Journal of Machine Learning Research 10 (2009) 301-340 Submitted 4/08; Revised 1/09; Published 2/09

Improving the Reliability of Causal Discovery from Small Data Sets Using Argumentation

Facundo Bromberg BROMBERG@CS.IASTATE.EDU
Dimitris Margaritis DMARG@CS.IASTATE.EDU
Dept. of Computer Science
Iowa State University
Ames, IA 50011

Editor: Constantin Aliferis

Abstract

We address the problem of improving the reliability of independence-based causal discovery algorithms that results from the execution of statistical independence tests on small data sets, which typically have low reliability. We model the problem as a knowledge base containing a set of independence facts that are related through Pearl's well-known axioms. Statistical tests on finite data sets may result in errors in these tests and inconsistencies in the knowledge base. We resolve these inconsistencies through the use of an instance of the class of defeasible logics called argumentation, augmented with a preference function, that is used to reason about and possibly correct errors in these tests. This results in a more robust conditional independence test, called an *argumentative independence test*. Our experimental evaluation shows clear positive improvements in the accuracy of argumentative over purely statistical tests. We also demonstrate significant improvements on the accuracy of causal structure discovery from the outcomes of independence tests both on sampled data from randomly generated causal models and on real-world data sets.

Keywords: independence-based causal discovery, causal Bayesian networks, structure learning, argumentation, reliability improvement

1. Introduction and Motivation

Directed graphical models, also called *Bayesian networks*, can be used to represent the probability distribution of a domain. This makes them a useful and important tool for machine learning where a common task is inference, that is, predicting the probability distribution of a variable of interest given some other knowledge, usually in the form of values of other variables in the domain. An additional use of Bayesian networks comes by augmenting them with causal semantics that represent cause and effect relationships in the domain. The resulting networks are called *causal*. An important problem is inferring the structure of these networks, a process that is sometimes called *causal discovery*, which can provide insights into the underlying data generation process.

Two major classes of algorithms exist for learning the structure of Bayesian networks. One class contains so-called *score-based* methods, which learn the structure by conducting a search in the space of all structures in an attempt to find the structure of maximum score. This score is usually penalized log-likelihood, for example, the Bayesian Information Criterion (BIC) or the (equivalent) Minimum Description Length (MDL). A second class of algorithms works by exploiting the fact that a causal Bayesian network implies the existence of a set of conditional independence statements between sets of domain variables. Algorithms in this class use the outcomes of a number of conditional independences to constrain the set of possible structures consistent with these to a singleton (if possible) and infer that structure as the only possible one. As such they are called *constraint-based* or *independence-based* algorithms. In this paper we address open problems related to the latter class of algorithms.

It is well-known that independence-based algorithms have several shortcomings. A major one has to do with the effect that unreliable independence information has on the their output. In general such independence information comes from two sources: (a) a domain expert that can provide his or her opinion on the validity of certain conditional independences among some of the variables, sometimes with a degree of confidence attached to them, and/or (b) statistical tests of independence, conducted on data gathered from the domain. As expert information is often costly and difficult to obtain, (b) is the most commonly used option in practice. A problem that occurs frequently however is that the data set available may be small. This may happen for various reasons: lack of subjects to observe (e.g., in medical domains), an expensive data-gathering process, privacy concerns and others. Unfortunately, the reliability of statistical tests significantly diminishes on small data sets. For example, Cochran (1954) recommends that Pearson's χ^2 independence test be deemed unreliable if more than 20% of the cells of the test's contingency table have an expected count of less than 5 data points. Unreliable tests, besides producing errors in the resulting causal model structure, may also produce cascading errors due to the way that independence-based algorithms work: their operation, including which test to evaluate next, typically depends on the outcomes of previous ones. Thus a single error in a statistical test can be propagated by the subsequent choices of tests to be performed by the algorithm, and finally when the edges are oriented. Therefore, an error in a previous test may have large (negative) consequences in the resulting structure, a property that is called *instability* in Spirtes et al. (2000). One possible method for addressing the effect of multiple errors in the construction of a causal model through multiple independence tests is the Bonferroni correction (Hochberg, 1988; Abdi, 2007), which works by dividing the type I error probability α of each test by the number of such tests evaluated during the entire execution of the causal learning algorithm. As a result, the collective type I error probability (of all tests evaluated) is α, that is, 0.05 typically. However, this may make the detection of true dependences harder, as now larger data sets would be required to reach the adjusted confidence threshold of each test. The types of adjustments that may be appropriate for each case to tests that may be dependent is an open problem and the subject of current research in statistics (Benjamini and Hochberg, 1995; Benjamini and Yekutieli, 2001; Storey, 2002).

In this paper we present and evaluate a number of methods for increasing the reliability of independence tests for small data sets. A result of this is the improvement in reliability of independence-based causal discovery algorithms that use these data sets, as we demonstrate in our experiments. We model this setting as a knowledge base whose contents are propositions representing conditional independences that may contain errors. Our main insight is to recognize that the outcomes of independence tests are not themselves independent but are constrained by the outcomes of other tests through Pearl's well-known properties of the conditional independence relation (Pearl, 1988; Dawid, 1979). These can therefore be seen as *integrity constraints* that can correct certain inconsistent test outcomes, choosing instead the outcome that can be inferred by tests that do not result in contradictions. We illustrate this by an example.

Example 1. *Consider an independence-based knowledge base that contains the following propositions, obtained through statistical tests on data.*

$$(\{0\} \perp\!\!\!\perp \{1\} \mid \{2\}) \tag{1}$$

$$(\{0\} \not\perp\!\!\!\perp \{3\} \mid \{2\}) \tag{2}$$

$$(\{0\} \perp\!\!\!\perp \{3\} \mid \{1,2\}) \tag{3}$$

where $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ *denotes conditional independence of the set of variables* $\mathbf{X}$ *with* $\mathbf{Y}$ *conditional on set* $\mathbf{Z}$*, and* $(\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ *denotes conditional dependence. Suppose that (3) is in fact wrong. Such an error can be avoided if there exists a constraint involving these independence propositions. For example, suppose that we also know that the following rule holds in the domain (this is an instance of an application of the Contraction and Decomposition axioms, described later in Section 2):*

$$(\{0\} \perp\!\!\!\perp \{1\} \mid \{2\}) \wedge (\{0\} \not\perp\!\!\!\perp \{3\} \mid \{2\}) \implies (\{0\} \not\perp\!\!\!\perp \{3\} \mid \{1,2\}). \tag{4}$$

Rule (4), together with independence proposition (1) and dependence proposition (2), contradict independence proposition (3), resulting in an inconsistent knowledge base. If Rule (4) and propositions (1) and (2) are accepted, then proposition (3) must be rejected (and its value reversed), correcting the error in this case. The framework presented in the rest of the paper provides a principled approach for resolving such inconsistencies.

The situation described in the previous example, while simple, illustrates the general idea that we will use in the rest of the paper: the set of independences and dependences used in a causal discovery algorithm form a potentially inconsistent knowledge base, and making use of general rules, derived from axioms and theorems that we know hold in the domain, helps us correct certain outcomes of statistical tests. In this way we will be able to improve the reliability of causal discovery algorithms that use them to derive causal models. To accomplish this we use the framework of *argumentation*, which provides a sound and elegant way of resolving inconsistencies in such knowledge bases, including ones that contain independences.

The rest of the paper is organized as follows. The next section introduces our notation and definitions. Section 3 presents the argumentation framework and its extension with preferences, and describes our approach for applying it to represent and reason in knowledge bases containing independence facts that may be inconsistent. Section 4 introduces the argumentative independence test, implemented by the top-down algorithm introduced in Section 5. We then present an approximation for the top-down algorithm in Section 6 that reduces its time complexity to polynomial. We experimentally evaluate our approach in Section 7, and conclude with a summary and possible directions of future research in Section 8. Most of the proofs are presented in detail in Appendices A and B, which contain proofs for the computability (termination) and the validity (no AIT test can return a dependence and an independence result at the same time) of AIT, respectively. Note that, as our main goal in this paper is to address the problem of robust causal learning and not necessarily to advance the theory of argumentation itself, our exposition in the rest of the paper is geared toward causality theorists and practitioners. As this community may be unfamiliar with the theory and methods of the argumentation framework, we have included a self-contained discussion that covers the basic definitions and theorems of argumentation theory in some detail.

2. Notation and Preliminaries

In this work we denote random variables with capitals (e.g., X,Y,Z) and sets of variables with bold capitals (e.g., $\mathbf{X},\mathbf{Y},\mathbf{Z}$). In particular, we denote by $\mathbf{V} = \{1,\ldots,n\}$ the set of all n variables in the domain, naming the variables by their indices in $\mathbf{V}$; for instance, we refer to the third variable in $\mathbf{V}$ simply by 3. We assume that all variables in the domain are discrete following a multinomial distribution or are continuous following a Gaussian distribution. We denote the data set by D and its size (number of data points) by N. We use the notation $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ to denote that the variables in set $\mathbf{X}$ are (jointly) independent of those in $\mathbf{Y}$ conditional on the values of the variables in $\mathbf{Z}$, for disjoint sets of variables $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$, while $(\mathbf{X} \not\!\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ denotes conditional dependence. For the sake of readability, we slightly abuse this notation and use $(X \perp\!\!\!\perp Y \mid Z)$ as shorthand for $(\{X\} \perp\!\!\!\perp \{Y\} \mid \{Z\})$.

A Bayesian network (**BN**) is a directed graphical model which represents the joint probability distribution over $\mathbf{V}$. Each node in the graph represents one of the random variables in the domain. The structure of the network implicitly represents a set of conditional independences on the domain variables. Given the structure of a BN, the set of independences implied by it can be identified by a process called *d-separation* (Pearl, 1988); the latter follows from the *local Markov property* that states that each node in the network is conditionally independent of all its non-descendants in the graph given its parents. All independences identified by d-separation are implied by the model structure. If, in addition, all remaining triplets $(\mathbf{X},\mathbf{Y},\mathbf{Z})$ correspond to dependencies, we say that the BN is *directed graph-isomorph* (abbreviated *DAG-isomorph*) or simply *causal* (as defined by Pearl, 1988). The concept of DAG-isomorphism is equivalent to a property called Faithfulness in Spirtes et al. (2000). A graph G is said to be *faithful* to some distribution if exactly those independences that exist in the distribution and no others are returned by the process of d-separation on G. In this paper we assume Faithfulness. For learning the structure of the Bayesian network of a domain we make use of the PC algorithm (Spirtes et al., 2000), which is only able to correctly identify the structure under the assumption of *causal sufficiency*. We therefore also assume causal sufficiency. A domain is causally sufficient if it does not contain any *hidden* or *latent* variables.

As mentioned above, independence-based algorithms operate by conducting a series of conditional independence queries. For these we assume that an *independence-query oracle* exists that is able to provide such information. This approach can be viewed as an instance of the statistical query oracle theory of Kearns and Vazirani (1994). In practice such an oracle does not exist, but can be implemented approximately by a statistical test evaluated on the data set (for example, this can be Pearson's conditional independence χ^2 (chi-square) test (Agresti, 2002), Wilk's G^2 test, a mutual information test etc.). In this work we used Wilk's G^2 test (Agresti, 2002). To determine conditional independence between two variables X and Y given a set $\mathbf{Z}$ from data, the statistical test G^2 (and many other independence tests based on hypothesis testing, for example, the χ^2 test) uses the values in the contingency table (a table containing the data point counts for each possible combination of the variables that participate in the test) to compute a *test statistic*. For a given value of the test statistic, the test then computes the likelihood of obtaining that or a more extreme value by chance under the *null hypothesis*, which in our case is that the two variables are conditionally independent. This likelihood, called the *p-value* of the test, is then returned. The p-value of a test equals the probability of falsely rejecting the null hypothesis (independence). Assuming that the p-value of a test is $p(X,Y \mid \mathbf{Z})$, the statistical test concludes independence if and only if $p(X,Y \mid \mathbf{Z})$ is greater than a threshold α, that is,

$$(X \perp\!\!\!\perp Y \mid \mathbf{Z}) \iff p(X,Y \mid \mathbf{Z}) > \alpha.$$

$$\boxed{\begin{array}{rl}
\textbf{(Symmetry)} & (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y} \mid \mathbf{Z}) \iff (\mathbf{Y}{\perp\!\!\!\perp}\mathbf{X} \mid \mathbf{Z}) \\
\textbf{(Decomposition)} & (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y}\cup\mathbf{W} \mid \mathbf{Z}) \implies (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X}{\perp\!\!\!\perp}\mathbf{W} \mid \mathbf{Z}) \\
\textbf{(Weak Union)} & (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y}\cup\mathbf{W} \mid \mathbf{Z}) \implies (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y} \mid \mathbf{Z}\cup\mathbf{W}) \\
\textbf{(Contraction)} & (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X}{\perp\!\!\!\perp}\mathbf{W} \mid \mathbf{Z}\cup\mathbf{Y}) \implies (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y}\cup\mathbf{W} \mid \mathbf{Z}) \\
\textbf{(Intersection)} & (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y} \mid \mathbf{Z}\cup\mathbf{W}) \wedge (\mathbf{X}{\perp\!\!\!\perp}\mathbf{W} \mid \mathbf{Z}\cup\mathbf{Y}) \implies (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y}\cup\mathbf{W} \mid \mathbf{Z})
\end{array}} \qquad (5)$$

$$\boxed{\begin{array}{rl}
\textbf{(Symmetry)} & (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y} \mid \mathbf{Z}) \iff (\mathbf{Y}{\perp\!\!\!\perp}\mathbf{X} \mid \mathbf{Z}) \\
\textbf{(Composition)} & (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X}{\perp\!\!\!\perp}\mathbf{W} \mid \mathbf{Z}) \implies (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y}\cup\mathbf{W} \mid \mathbf{Z}) \\
\textbf{(Decomposition)} & (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y}\cup\mathbf{W} \mid \mathbf{Z}) \implies (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X}{\perp\!\!\!\perp}\mathbf{W} \mid \mathbf{Z}) \\
\textbf{(Intersection)} & (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y} \mid \mathbf{Z}\cup\mathbf{W}) \wedge (\mathbf{X}{\perp\!\!\!\perp}\mathbf{W} \mid \mathbf{Z}\cup\mathbf{Y}) \implies (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y}\cup\mathbf{W} \mid \mathbf{Z}) \\
\textbf{(Weak Union)} & (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y}\cup\mathbf{W} \mid \mathbf{Z}) \implies (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y} \mid \mathbf{Z}\cup\mathbf{W}) \\
\textbf{(Contraction)} & (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X}{\perp\!\!\!\perp}\mathbf{W} \mid \mathbf{Z}\cup\mathbf{Y}) \implies (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y}\cup\mathbf{W} \mid \mathbf{Z}) \\
\textbf{(Weak Transitivity)} & (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X}{\perp\!\!\!\perp}\mathbf{Y} \mid \mathbf{Z}\cup\gamma) \implies (\mathbf{X}{\perp\!\!\!\perp}\gamma \mid \mathbf{Z}) \vee (\gamma{\perp\!\!\!\perp}\mathbf{Y} \mid \mathbf{Z}) \\
\textbf{(Chordality)} & (\alpha{\perp\!\!\!\perp}\beta \mid \gamma\cup\delta) \wedge (\gamma{\perp\!\!\!\perp}\delta \mid \alpha\cup\beta) \implies (\alpha{\perp\!\!\!\perp}\beta \mid \gamma) \vee (\alpha{\perp\!\!\!\perp}\beta \mid \delta)
\end{array}} \qquad (6)$$

Common values in statistics for α are 0.05 and 0.01, corresponding to *confidence thresholds* $(1-\alpha)$ of 0.95 and 0.99 respectively. The value 0.10 for α is also sometimes used, depending on the application, while values as low as 0.005 and 0.001 are sometimes used for structure learning.

The conditional independences and dependences of a domain are connected through a set of general rules, introduced in Pearl (1988) and shown boxed in Eq. (5). These can be seen as constraints in a meta-space representing all possible independences in the domain. More specifically, let us imagine a meta-space of binary variables, each corresponding to the truth value of the independence of a triplet $(\mathbf{X}, \mathbf{Y} \mid \mathbf{Z})$ (e.g., `true` for independence and `false` for dependence). Each point in this space corresponds to a conditional independence assignment to all possible triplets in the domain. In this conceptual space not all points are tenable; in particular the set of rules of Eq. (5) constrain the truth values of independences corresponding to triplets. For domains for which there exists a faithful Bayesian network a more relaxed set of properties hold, shown boxed in Eq. (6) where α, β, γ and δ correspond to single variables. In both sets of axioms, the property of Intersection holds if the probability distribution of the domain is positive, meaning that every assignment to all variables in the domain has a non-zero probability. Eq. (6) were first introduced by Dawid (1979) in a slightly different form and independently re-discovered by Pearl and Paz (1985).

Note that the axioms of Eq. (5) are necessarily incomplete; Studený (1991) showed that there is no finite axiomatization of the conditional independence relation in general. The implication of this is that there may be some inconsistencies involving some set of independences and dependences that no method can detect and resolve.

In the next section we describe the argumentation framework, which allows one to make beneficial use of these constraints. This is followed by its application to our problem of answering independence queries from knowledge bases that contain sets of potentially inconsistent independence propositions.

3. The Argumentation Framework

There exist two major approaches for reasoning with information contained in inconsistent knowledge bases such as those containing independence statements that were described in the previous section. These two distinct approaches correspond to two different attitudes: One is to resolve the inconsistencies by removing a subset of propositions such that the resulting KB becomes consistent; this is called *belief revision* in the literature (Gärdenfors, 1992; Gärdenfors and Rott, 1995; Shapiro, 1998; Martins, 1992). A potential shortcoming (Shapiro, 1998) of belief revision stems from the fact that it removes propositions, which discards potentially valuable information. In addition, an erroneous modification of the KB (such as the removal of a proposition) may have unintended negative consequences if later more propositions are inserted in the KB. A second approach to inconsistent KBs is to allow inconsistencies but to use rules that may be possibly contained in it to deduce which truth value of a proposition query is "preferred" in some way. One instance of this approach is *argumentation* (Dung, 1995; Loui, 1987; Prakken, 1997; Prakken and Vreeswijk, 2002), which is a sound approach that allows inconsistencies but uses a proof procedure that is able to deduce (if possible) that one of the truth values of certain propositions is preferred over its negation. Argumentation is a reasoning model that belongs to the broader class of defeasible logics (Pollock, 1992; Prakken, 1997). Our approach uses the argumentation framework of Amgoud and Cayrol (2002) that considers preferences over arguments, extending Dung's more fundamental framework (Dung, 1995). Preference relations give an extra level of specificity for comparing arguments, allowing a more refined form of selection between conflicting propositions. Preference-based argumentation is presented in more detail in Section 3.2.

We proceed now to describe the argumentation framework.

Definition 1. *An* argumentation framework *is a pair $\langle \mathcal{A}, \mathcal{R} \rangle$, where $\mathcal{A}$ is a set of arguments and $\mathcal{R}$ is a binary relation representing a defeasibility relationship between arguments, that is, $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$. $(a,b) \in \mathcal{R}$ or equivalently "a $\mathcal{R}$ b" reads that argument a defeats argument b. We also say that a and b are in conflict.*

An example of the defeat relation $\mathcal{R}$ is *logical defeat*, which occurs when an argument contradicts another logically.

The elements of the argumentation framework are not propositions but *arguments*. Given a potentially inconsistent knowledge base $\mathcal{K} = \langle \Sigma, \Psi \rangle$ with a set of propositions Σ and a set of inference rules Ψ, arguments are defined formally as follows.

Definition 2. *An* argument *over knowledge base $\langle \Sigma, \Psi \rangle$ is a pair (H,h) where $H \subseteq \Sigma$ such that:*

- *H is consistent,*
- *$H \vdash_{\Psi} h$,*
- *H is minimal (with respect to set inclusion).*

H is called the support *and h the* conclusion *or* head *of the argument.*

In the above definition $\vdash_{\Psi}$ stands for classical logical inference over the set of inference rules Ψ. Intuitively an argument (H,h) can be thought as an "if-then" rule, that is, "if H then h." In inconsistent knowledge bases two arguments may contradict or *defeat* each other. The defeat relation is defined through the *rebut* and *undercut* relations, defined as follows.

Definition 3. *Let (H_1,h_1), (H_2,h_2) be two arguments.*

- (H_1,h_1) rebuts (H_2,h_2) *iff* $h_1 \equiv \neg h_2$.

Algorithm 1 Recursive computation of acceptable arguments: $Acc_{\mathcal{R}} = \mathcal{F}(\mathcal{A}, \mathcal{R}, S)$

```
1: $S' \longleftarrow S \cup \{a \in \mathcal{A} \mid a \text{ is defended by } S\}$
2: if $S = S'$ then
3:     return $S'$
4: else
5:     return $\mathcal{F}(\mathcal{A}, \mathcal{R}, S')$
```

- (H_1, h_1) undercuts (H_2, h_2) *iff* $\exists h \in H_2$ *such that* $h \equiv \neg h_1$.

If (H_1, h_1) *rebuts or undercuts* (H_2, h_2) *we say that* (H_1, h_1) defeats (H_2, h_2).

(The symbol "$\equiv$" stands for logical equivalence.) In other words, $(H_1, h_1)\ \mathcal{R}\ (H_2, h_2)$ if and only if (H_1, h_1) *rebuts* or *undercuts* (H_2, h_2).

The objective of argumentation is to decide on the acceptability of a given argument. There are three possibilities: an argument can be accepted, rejected, or neither. This partitions the space of arguments $\mathcal{A}$ in three classes:

- The class $Acc_{\mathcal{R}}$ of *acceptable arguments*. Intuitively, these are the "good" arguments. In the case of an inconsistent knowledge base, these will be inferred from the knowledge base.

- The class $Rej_{\mathcal{R}}$ of *rejected arguments*. These are the arguments defeated by acceptable arguments. When applied to an inconsistent knowledge base, these will not be inferred from it.

- The class $Ab_{\mathcal{R}}$ of arguments *in abeyance*. These arguments are neither accepted nor rejected.

The semantics of acceptability proposed by Dung (1995) dictates that an argument should be accepted if it is not defeated, or if it is defended by acceptable arguments, that is, each of its defeaters is itself defeated by an acceptable argument. This is formalized in the following definitions.

Definition 4. *Let* $\langle \mathcal{A}, \mathcal{R} \rangle$ *be an argumentation framework, and* $S \subseteq \mathcal{A}$. *An argument* a *is* defended *by* S *if and only if* $\forall b$, *if* $(b\ \mathcal{R}\ a)$ *then* $\exists c \in S$ *such that* $(c\ \mathcal{R}\ b)$.

Dung characterizes the set of acceptable arguments by a monotonic function $\mathcal{F}$, that is, $\mathcal{F}(S) \subseteq \mathcal{F}(S \cup T)$ for some S and T. Given a set of arguments $S \subseteq \mathcal{A}$ as input, $\mathcal{F}$ returns the set of all arguments defended by S:

Definition 5. *Let* $S \subseteq \mathcal{A}$. *Then* $\mathcal{F}(S) = \{a \in \mathcal{A} \mid a \text{ is defended by } S\}$.

Slightly overloading our notation, we define $\mathcal{F}(\varnothing)$ to contain the set of arguments that are not defeated by any argument in the framework.

Definition 6. $\mathcal{F}(\varnothing) = \{a \in \mathcal{A} \mid a \text{ is not defeated by any argument in } \mathcal{A}\}$.

Dung proved that the set of acceptable arguments is the least fix-point of $\mathcal{F}$, that is, the smallest set S such that $\mathcal{F}(S) = S$.

Theorem 7 (Dung 1995)**.** *Let* $\langle \mathcal{A}, \mathcal{R} \rangle$ *be an argumentation framework. The set of acceptable arguments* $Acc_{\mathcal{R}}$ *is the least fix-point of the function* $\mathcal{F}$.

Dung also showed that if the argumentation framework $\langle \mathcal{A}, \mathcal{R} \rangle$ is finitary, that is, for each argument A there are finitely many arguments that defeat A, the least fix-point of function $\mathcal{F}$ can be obtained by iterative application of $\mathcal{F}$ to the empty set. We can understand this intuitively: From our semantics of acceptability it follows that all arguments in $\mathcal{F}(\varnothing)$ are accepted. Also, every argument in $\mathcal{F}(\mathcal{F}(\varnothing))$ must be acceptable as well since each of its arguments is defended by acceptable arguments. This reasoning can be applied recursively until a fix-point is reached. This happens when the arguments in S cannot be used to defend any other argument not in S, that is, no additional argument is accepted. This suggests a simple algorithm for computing the set of acceptable arguments. Algorithm 1 shows a recursive procedure for this, based on the above definition. The algorithm takes as input an argumentation framework $\langle \mathcal{A}, \mathcal{R} \rangle$ and the set S of arguments found acceptable so far, that is, $S = \varnothing$ initially.

Let us illustrate these ideas with an example.

Example 2. *Let $\langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework defined by $\mathcal{A} = \{a, b, c\}$ and $\mathcal{R} = \{(a,b), (b,c)\}$. The only argument that is not defeated is a, and therefore $\mathcal{F}(\varnothing) = \{a\}$. Argument b is defeated by the acceptable argument a, so b cannot be defended and is therefore rejected, that is, $b \in Rej_{\mathcal{R}}$. Argument c, though defeated by b, is defended by (acceptable argument) a which defeats b, so c is acceptable. The set of acceptable arguments is therefore $Acc_{\mathcal{R}} = \{a, c\}$ and the set of rejected arguments is $Rej_{\mathcal{R}} = \{b\}$.*

The **bottom-up** approach of Algorithm 1 has the disadvantage that it requires the computation of all acceptable arguments to answer the acceptability status of a single one. In practice, and in particular in the application of argumentation to independence tests, the entire set of acceptable arguments is rarely needed. An alternative is to take a top-down approach (Amgoud and Cayrol, 2002; Dung, 1995; Toni and Kakas, 1995; Kakas and Toni, 1999) that evaluate the acceptability of some input argument by evaluating (recursively) the acceptability of its attackers. Below we present an alternative algorithm, called the **top-down algorithm**, for deciding the acceptability of an input argument. This algorithm is a version of the *dialog tree* algorithm of Amgoud and Cayrol (2002), where details unnecessary for the current exposition are not shown. This algorithm is provably equivalent to Algorithm 1 (whenever it is given the same input it is guaranteed to produce the same output), but it is considerably more efficient (as shown later in Section 5.2). We sketch the algorithm here and show a concrete version using the preference-based argumentation framework in Section 3.2.

Given an input argument a, the top-down algorithm employs a goal-driven approach for answering whether a is accepted or not. Its operation is guided by the same acceptability semantics as those used for Algorithm 1. Let us denote the predicates $A(a) \equiv (a \in Acc_{\mathcal{R}})$, $R(a) \equiv (a \in Rej_{\mathcal{R}})$, and $Ab(a) \equiv (a \in Ab_{\mathcal{R}})$. Then, the acceptability semantics are as follows.

Algorithm 2 Top-down computation of acceptable arguments: *top-down*$(\mathcal{A}, \mathcal{R}, a)$

```
1: defeaters ← set of arguments in $\mathcal{A}$ that defeat $a$ according to $\mathcal{R}$.
2: for $d \in$ defeaters do
3:     if top-down$(\mathcal{A}, \mathcal{R}, d)$ = accepted then
4:         return rejected
5: return accepted
```

(Acceptance) A node is accepted iff it has no defeaters or all its defeaters are rejected:

$$A(a) \iff \forall b \in \mathit{defeaters}(a), R(b).$$

(Rejection) A node is rejected iff at least one of its defeaters is accepted:

$$R(a) \iff \exists b \in \mathit{defeaters}(a), A(b). \tag{7}$$

(Abeyance) A node is in abeyance iff its not accepted nor rejected:

$$Ab(a) \iff \neg A(a) \wedge \neg R(a).$$

The logic of these equations can be easily implemented with a recursive algorithm, shown in Algorithm 2. The algorithm, given some input argument *a*, loops over all defeaters of *a* and responds `rejected` if any of its defeaters is accepted (line 4). If execution reaches the end of the loop at line 5 then that means that none of its defeaters was accepted, and thus the algorithm accepts the input argument *a*. We can represent the execution of the top-down algorithm graphically by a tree that contains *a* at the root node, and all the defeaters of a node as its children. A leaf is reached when a node has no defeaters. In that case the loop contains no iterations and line 5 is reached trivially.

Unfortunately, the top-down algorithm, as shown in Algorithm 2, will fail to terminate when a node is in abeyance. This is clear from the following lemma (proved formally in Appendix A but reproduced here to aid our intuition).

Lemma 8. *For every argument a,*

$$Ab(a) \implies \exists b \in \mathit{attackers}(a), Ab(b).$$

(An attacker is a type of defeater; it is explained in detail in the next section. For the following discussion the reader can substitute “attacker” with “defeater” in the lemma above.) From this lemma we can see that, if an argument is in abeyance, its set of defeaters must contain an argument in abeyance and thus the recursive call of the top-down algorithm will never terminate, as there will always be another defeater in abeyance during each call. While there are ways to overcome this difficulty in the general case, we can prove that using the preference-based argumentation framework (presented later in the paper) and for the particular preference relation introduced for deciding on independence tests (c.f. Section 3.3), no argument can be in abeyance and thus the top-down algorithm always terminates. A formal proof of this is presented later in Section 5.

We conclude the section by proving that the top-down algorithm is equivalent to the bottom-up algorithm of Algorithm 1 that is, given the same input as Algorithm 1 it is guaranteed to produce the same output. The proof assumes no argument is in abeyance. This assumption is satisfied for argumentation in independence knowledge bases (c.f. Theorem 20, Section 5).

Theorem 9. *Let a be an argument in the argumentation framework $\langle \mathcal{A}, \mathcal{R} \rangle$, and let $\mathcal{F}$ be the set of acceptable arguments output by Algorithm 1. Assuming a is not in abeyance,*

$$\mathit{top\text{-}down}(\mathcal{A}, \mathcal{R}, a) = \texttt{accepted} \iff a \in \mathcal{F} \tag{8}$$

$$\mathit{top\text{-}down}(\mathcal{A}, \mathcal{R}, a) = \texttt{rejected} \iff a \notin \mathcal{F}. \tag{9}$$

Proof According to Theorem 7, the fix point of function $\mathcal{F}$ returned by Algorithm 1 contains the set of arguments considered acceptable by the acceptability semantics of Dung. As the top-down algorithm is a straightforward implementation of Dung's acceptability semantics expressed by Eq. (7), the double implication of Eq. (8) must follow. To prove Eq. (9) we can prove the equivalent expression with both sides negated, that is,

$$\textit{top-down}(\mathcal{A}, \mathcal{R}, a) \neq \texttt{rejected} \iff a \in \mathcal{F}.$$

Since a is not in abeyance, if the top-down algorithm does not return `rejected` it must return `accepted`. The double implication is thus equivalent to Eq. (8), which was proved true. ■

3.1 Argumentation in Independence Knowledge Bases

We can now apply the argumentation framework to our problem of answering queries from knowledge bases that contain a number of potentially inconsistent independences and dependencies and a set of rules that express relations among them.

Definition 10. *An* independence knowledge base ***(IKB)*** *is a knowledge base $\langle \Sigma, \Psi \rangle$ such that its set of propositions Σ contains independence propositions of the form $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ or $(\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$ for $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ disjoint subsets of $\mathbf{V}$, and its set of inference rules Ψ is either the general set of axioms shown in Eq. (5) or the specific set of axioms shown in Eq. (6).*

For IKBs, the set of arguments $\mathcal{A}$ is obtained in two steps. First, for each proposition $\sigma \in \Sigma$ (independence or dependence) we add to $\mathcal{A}$ the argument $(\{\sigma\}, \sigma)$. This is a valid argument according to Definition 2 since its support $\{\sigma\}$ is (trivially) consistent, it (trivially) implies the head σ, and it is minimal (the pair $(\varnothing, \sigma)$ is not a valid argument since $\varnothing$ is equivalent to the proposition `true` which does not entail σ in general). We call arguments of the form $(\{\sigma\}, \sigma)$ *propositional arguments* since they correspond to single propositions. The second step in the construction of the set of arguments $\mathcal{A}$ concerns rules. Based on the chosen set of axioms (general or directed) we construct an alternative, logically equivalent set of rules Ψ', each member of which is *single-headed*, that is, contains a single proposition as the consequent, and *decomposed*, that is, each of its propositions is an independence statement over single variables (the last step is justified by the fact that typical algorithms for causal learning never produce nor require the evaluation of independence between sets).

To construct the set of single-headed rules we consider, for each axiom, all possible contrapositive versions of it that have a single head. To illustrate, consider the Weak Transitivity axiom

$$(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \gamma) \implies (\mathbf{X} \perp\!\!\!\perp \gamma \mid \mathbf{Z}) \vee (\gamma \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})$$

from which we obtain the following set of single-headed rules:

$$\begin{aligned}
(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \gamma) \wedge (\mathbf{X} \not\perp\!\!\!\perp \gamma \mid \mathbf{Z}) &\implies (\gamma \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \\
(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \gamma) \wedge (\gamma \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) &\implies (\mathbf{X} \perp\!\!\!\perp \gamma \mid \mathbf{Z}) \\
(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \gamma) \wedge (\gamma \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \not\perp\!\!\!\perp \gamma \mid \mathbf{Z}) &\implies (\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \\
(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\gamma \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X} \not\perp\!\!\!\perp \gamma \mid \mathbf{Z}) &\implies (\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z} \cup \gamma).
\end{aligned}$$

To obtain decomposed rules we apply the Decomposition axiom to every single-headed rule to produce only propositions over singletons. To illustrate, consider the Intersection axiom:

$$(\mathbf{X}\perp\!\!\!\perp\mathbf{Y} \mid \mathbf{Z}\cup\mathbf{W}) \wedge (\mathbf{X}\perp\!\!\!\perp\mathbf{W} \mid \mathbf{Z}\cup\mathbf{Y}) \implies (\mathbf{X}\perp\!\!\!\perp\mathbf{Y}\cup\mathbf{W} \mid \mathbf{Z}).$$

In the above the consequent coincides with the antecedent of the Decomposition axiom, and we thus replace the Intersection axiom with a decomposed version:

$$(\mathbf{X}\perp\!\!\!\perp\mathbf{Y} \mid \mathbf{Z}\cup\mathbf{W}) \wedge (\mathbf{X}\perp\!\!\!\perp\mathbf{W} \mid \mathbf{Z}\cup\mathbf{Y}) \implies (\mathbf{X}\perp\!\!\!\perp\mathbf{Y} \mid \mathbf{Z}) \wedge (\mathbf{X}\perp\!\!\!\perp\mathbf{W} \mid \mathbf{Z}).$$

Finally, note that it is easy to show that this rule is equivalent to two single-headed rules, one implying $(\mathbf{X}\perp\!\!\!\perp\mathbf{Y} \mid \mathbf{Z})$ and the other implying $(\mathbf{X}\perp\!\!\!\perp\mathbf{W} \mid \mathbf{Z})$.

The result of the application of the above procedures is a set of single-headed, decomposed rules Ψ'. We construct, for each such rule $(\Phi_1 \wedge \Phi_2 \ldots \wedge \Phi_n \implies \varphi) \in \Psi'$ and for each subset of Σ that matches exactly the set of antecedents, that is, each subset $\{\varphi_1, \varphi_2 \ldots, \varphi_n\}$ of Σ such that $\Phi_1 \equiv \varphi_1, \Phi_2 \equiv \varphi_2 \ldots \Phi_n \equiv \varphi_n$, the argument $(\{\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n\}, \varphi)$, and add it to $\mathcal{A}$.[1]

IKBs can be augmented with a set of preferences that allow one to take into account the reliability of each test when deciding on the truth value of independence queries. This is described in the next section.

3.2 Preference-based Argumentation Framework

Following Amgoud and Cayrol (2002), we now refine the argumentation framework of Dung (1995) for cases where it is possible to define a preference order Π over arguments.

Definition 11. *A* preference-based argumentation framework ***(PAF)*** *is a triplet $\langle \mathcal{A}, \mathcal{R}, \Pi \rangle$ where $\mathcal{A}$ is a set of arguments, $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ is a binary relation representing a defeat relationship between pairs of arguments, and Π is a (partial or total) order over $\mathcal{A}$.*

For the case of inconsistent knowledge bases, preference Π over arguments follows the preference π over their support, that is, stronger support implies a stronger argument, which is given as a partial or total order over sets of propositions. Formally:

Definition 12. *Let $\mathcal{K} = \langle \Sigma, \Psi \rangle$ be a knowledge base, π be a (partial or total) order on subsets of Σ and (H,h), (H',h') two arguments over $\mathcal{K}$. Argument (H,h) is* π-preferred *to (H',h') (denoted $(H,h) \gg_\pi (H',h')$) if and only if H is preferred to H' with respect to π.*

In what follows we overload our notation by using π to denote either the ordering over arguments or over their supports.

An important sub-class of preference relations is the *strict* and *transitive* preference relation, defined as follows.

Definition 13. *We say that preference relation π over arguments is* strict *if the order of arguments induced by it is strict and total, that is, for every pair of arguments a and b,*

$$\big(a \gg_\pi b\big) \iff \neg\big(b \gg_\pi a\big).$$

1. This is equivalent to propositionalizing the set of rules, which are first-order (the rules of Eqs. (5) and (6) are universally quantified over all sets of variables, and thus are the rules in Ψ'). As this may be expensive (exponential in the number of propositions), in practice it is not implemented in this way; instead, appropriate rules are matched on the fly during the argumentation inference process.

Definition 14. *We say that preference relation* π *over arguments is* transitive *if, for every three arguments a, b and c,*

$$\big(a \gg_\pi b\big) \wedge \big(b \gg_\pi c\big) \implies \big(a \gg_\pi c\big).$$

The importance of the properties of strictness and transitivity will become clear later when we talk about the correctness of the argumentative independence test (defined later in Section 4).

We now introduce the concept of *attack* relation, a combination of the concepts of defeat and preference relation.

Definition 15. *Let* $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ *be a PAF, and a,* $b \in \mathcal{A}$ *be two arguments. We say b* attacks *a if and only if b* $\mathcal{R}$ *a and* $\neg(a \gg_\pi b)$.

We can see that the attack relation is a special case of the defeat relation and therefore the same conclusions apply; in particular Theorem 7, which allows us to compute the set of acceptable arguments of a PAF using Algorithm 1 or Algorithm 2.

In Sections 3.3 and 4 below, we apply these ideas to construct an approximation to the independence-query oracle that is more reliable than a statistical independence test.

3.3 Preference-based Argumentation in Independence Knowledge Bases

We now describe how to apply the preference-based argumentation framework of Section 3.2 to improve the reliability of conditional independence tests conducted on a (possibly small) data set. A preference-based argumentation framework has three components. The first two, namely $\mathcal{A}$ and $\mathcal{R}$, are identical to the general argumentation framework. We now describe how to construct the third component, namely the preference order π over subsets H of Σ, in IKBs. We define it using a belief estimate $\nu(H)$ that all propositions in H are correct,

$$H \gg_\pi H' \Longleftrightarrow \nu(H) > \nu(H') \vee \big[\nu(H) = \nu(H') \wedge f(H, H')\big]. \tag{10}$$

That is, H is preferred over H' if and only if its belief of correctness is higher than that of H' or, in the case that these beliefs are equal, we break the tie using predicate f. For that we require that

$$\forall H, H' \subseteq \mathcal{A}, \text{ such that } H \neq H', \;\; f(H, H') = \neg f(H', H). \tag{11}$$

In addition, we require that f be transitive, that is, $f(H, H') \wedge f(H', H'') \Longrightarrow f(H, H'')$. This implies that the preference relation π is transitive, which is a necessary condition for proving a number of important theorems in Appendix A. In our implementation we resolved ties by assuming an arbitrary order of the variables in the domain, determined at the beginning of the algorithm and maintained fixed during its entire execution. Based on this ordering, $f(H, H')$ resolved ties by the lexicographic order of the variables in H and H'. By this definition, our f is both non-commutative and transitive.

Before we define $\nu(H)$ we first show that π, as defined by Eqs. (10) and (11) and for any definition of $\nu(H)$, satisfies two important properties, namely strictness (Definition 13) and transitivity (Definition 14). We do this in the following two lemmas.

Lemma 16. *The preference relation for independence knowledge bases defined by Equations (10) and (11) is strict.*

Proof

$$\begin{array}{lll}
H \gg_\pi H' & & \\
\iff & \nu(H) > \nu(H') \vee \left[\nu(H) = \nu(H') \wedge f(H,H')\right] & \textbf{[By Eq. (10)]} \\
\iff & \nu(H) \geq \nu(H') \wedge \left[\nu(H) > \nu(H') \vee f(H,H')\right] & \textbf{[Distributivity of } \vee \textbf{ over } \wedge\textbf{]} \\
\iff & \neg\left\{\nu(H') > \nu(H) \vee \left[\nu(H') \geq \nu(H) \wedge f(H',H)\right]\right\} & \textbf{[Double negation and Eq. (11)]} \\
\iff & \neg\left\{\left[\nu(H') > \nu(H) \vee \nu(H') \geq \nu(H)\right] \wedge \left[\nu(H') > \nu(H) \vee f(H',H)\right]\right\} & \\
\iff & \neg\left\{\nu(H') \geq \nu(H) \wedge \left[\nu(H') > \nu(H) \vee f(H',H)\right]\right\} & \\
\iff & \neg\left\{\left[\nu(H') > \nu(H) \vee \nu(H') = \nu(H)\right] \wedge \left[\nu(H') > \nu(H) \vee f(H',H)\right]\right\} & \\
\iff & \neg\left\{\nu(H') > \nu(H) \vee \left[\nu(H') = \nu(H) \wedge f(H',H)\right]\right\} & \textbf{[Common factor } \nu(H') > \nu(H)\textbf{]} \\
\iff & \neg(H' \gg_\pi H) & \textbf{[Again by Eq. (10)]}
\end{array}$$

■

Lemma 17. *The preference relation defined by Equations (10) and (11) is transitive.*

Proof

$$\begin{array}{lll}
H \gg_\pi J \wedge J \gg_\pi K & & \\
\iff & \Big\{\nu(H) > \nu(J) \vee \left[\nu(H) = \nu(J) \wedge f(H,J)\right]\Big\} & \\
 & \wedge \Big\{\nu(J) > \nu(K) \vee \left[\nu(J) = \nu(K) \wedge f(J,K)\right]\Big\} & \textbf{[By Eq. (10)]} \\
\iff & \left[\nu(H) > \nu(J) \wedge \nu(J) > \nu(K)\right] & \textbf{[Case A]} \\
 & \vee \left[\nu(H) > \nu(J) \wedge \nu(J) = \nu(K) \wedge f(J,K)\right] & \textbf{[Case B]} \\
 & \vee \left[\nu(H) = \nu(J) \wedge f(H,J) \wedge \nu(J) > \nu(K)\right] & \textbf{[Case C]} \\
 & \vee \left[\nu(H) = \nu(J) \wedge f(H,J) \wedge \nu(J) = \nu(K) \wedge f(J,K)\right] & \textbf{[Case D]}
\end{array}$$

To complete the proof we show that each of the cases A, B, C and D implies $H \gg_\pi K$.

(Case A) $\nu(H) > \nu(J) \wedge \nu(J) > \nu(K) \Longrightarrow \nu(H) > \nu(K) \Longrightarrow H \gg_\pi K$.

(Case B) $\nu(H) > \nu(J) \wedge \nu(J) = \nu(K) \wedge f(J,K) \Longrightarrow \nu(H) > \nu(K) \Longrightarrow H \gg_\pi K$.

(Case C) $\nu(H) = \nu(J) \wedge f(H,J) \wedge \nu(J) > \nu(K) \Longrightarrow \nu(H) > \nu(K) \Longrightarrow H \gg_\pi K$.

(Case D)

$$\begin{array}{lll}
\nu(H) = \nu(J) \wedge f(H,J) \wedge \nu(J) = \nu(K) \wedge f(J,K) & \Longrightarrow & \nu(H) = \nu(K) \wedge f(H,K) \\
 & \Longrightarrow & H \gg_\pi K,
\end{array}$$

due to the transitivity of predicate f.

■

We now return to the computation of $\nu(H)$. We estimate the belief $\nu(H)$ that a set of propositions H is correct by assuming independence among these propositions.[2] Overloading notation and denoting by $\nu(h)$ the probability of an individual proposition h being correct, the probability of all elements in H being correct under this assumption of independence is

$$\nu(H) = \prod_{h \in H} \nu(h). \tag{12}$$

The belief that a proposition stating independence is correct can be computed in different ways, depending on the particular choice of independence oracle chosen. In this paper we use Wilk's G^2 test, but the resulting belief can be easily adapted to any other independence oracle that produces p-values. We hope that the following discussion serves as a starting point for others to adapt it to other types of independence oracles.

As discussed in Section 2, the p-value $p(X,Y \mid \mathbf{Z})$ computed by this test is the probability of error in rejecting the null hypothesis (conditional independence in our case) and assuming that X and Y are dependent. Therefore, the probability of a test returning dependence of being correct is

$$\nu_D(X \not\perp\!\!\!\perp Y \mid \mathbf{Z}) = 1 - p(X,Y \mid \mathbf{Z})$$

where the subscript D indicates that this expression is valid only for dependencies. Formally, the error of falsely rejecting the null hypothesis is called a *type I error*. To determine the preference of a test returning independence we can, in principle, use this procedure symmetrically: use the probability of error in falsely accepting the null hypothesis (again, this is conditional independence), called a *type II error*, which we denote by $\beta(X,Y \mid \mathbf{Z})$. In this case we can define the preference of independence $(X \perp\!\!\!\perp Y \mid \mathbf{Z})$ as the probability of correctly assuming independence by

$$\nu_I(X \perp\!\!\!\perp Y \mid \mathbf{Z}) = 1 - \beta(X,Y \mid \mathbf{Z})$$

where again the subscript I indicates that it is valid only for independences. Unfortunately value of β cannot be obtained without assumptions, because it requires the computation of the probability of the test statistic under the hypothesis of dependence, and there are typically an infinite number of dependent models. In statistical applications, the β value is commonly approximated by assuming one particular dependence model if prior knowledge about that is available. In the absence of such information however in this paper we estimate it using a heuristic function of the p-value, assuming the following heuristic constraints on β:

$$\beta(X,Y \mid \mathbf{Z}) = \begin{cases} 1 & \text{if } p(X,Y \mid \mathbf{Z}) = 0 \\ \alpha - \frac{\alpha}{2+|\mathbf{Z}|} & \text{if } p(X,Y \mid \mathbf{Z}) = 1 \\ \alpha & \text{if } p(X,Y \mid \mathbf{Z}) = \alpha. \end{cases}$$

The first constraint (for $p(X,Y \mid \mathbf{Z}) = 0$) corresponds to the intuition that when the p-value of the test is close to 0, the test statistic is very far from its value under the model that assumes independence, and thus we would give more preference to the "dependence" decision. The intuition for

2. The assumption of independence is a heuristic, and is made mainly due to the difficulty of determining the dependence between two or more statistical tests evaluated on the same data set. Other possible ways of defining the preference of a set of propositions are possible. The problem of dealing with multiple tests is an open problem and an area of active research in statistics; see Section 1 for a discussion.

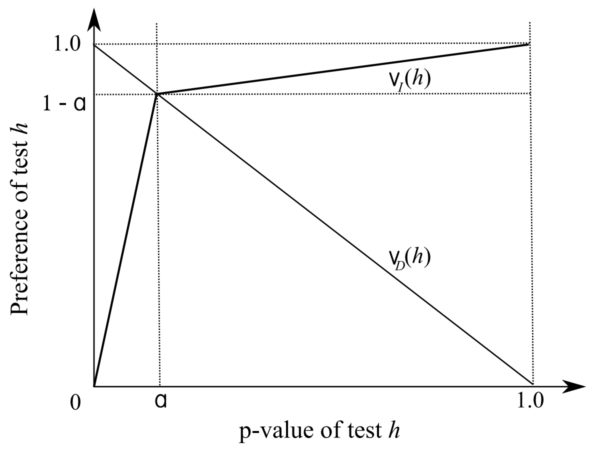

Figure 1: Preference functions $\nu_I(h)$ and $\nu_D(h)$ for statements of independence and dependence respectively, as functions of the p-value of test h.

the second case $(p(X,Y \mid \mathbf{Z}) = 1)$ is reversed—when the value of the statistic is very close to the expected one under independence then independence is preferred. The value of the second case is tempered by the number of variables in the conditioning set. This reflects the practical consideration that, as the number $2+|\mathbf{Z}|$ of variables involved in the test increases, given a fixed data set, the discriminatory power of the test diminishes as $|\mathbf{Z}| \to \infty$. The third case causes the two functions ν_I and ν_D to intersect at p-value α. This is due to fairness: in the absence of non-propositional arguments (i.e., in the absence of inference rules in our knowledge base), the independence decisions of the argumentation framework should match those of the purely statistical tests, that is, "dependence" if and only if (p-value $\leq \alpha$). If instead we chose a different intersection point, then the resulting change in the outcome of tests may have been simply due to bias in the independence decision that favors dependence or independence, that is, equivalent to an arbitrary change of the threshold of the statistical test, and the comparison of the statistical and the new test based on argumentation would not be a fair one. The remaining values of β are approximated by linear interpolation among the above points. The result is summarized in Fig. 1, which depicts preference functions ν_D and ν_I with respect to the p-value of the corresponding statistical test.

Let us illustrate how the preference-based argumentation can be used to resolve the inconsistencies of Example 1.

Example 3. *In example 1 we considered an IKB with the following propositions*

$$(0 \perp\!\!\!\perp 1 \mid 2) \tag{13}$$
$$(0 \not\perp\!\!\!\perp 3 \mid 2) \tag{14}$$
$$(0 \perp\!\!\!\perp 3 \mid \{1,2\}) \tag{15}$$
$$(0 \perp\!\!\!\perp 1 \mid 2) \wedge (0 \not\perp\!\!\!\perp 3 \mid 2) \implies (0 \not\perp\!\!\!\perp 3 \mid \{1,2\}). \tag{16}$$

Following the IKB construction procedure described in the previous section, propositions (13), (14) and (15) correspond to the following arguments, respectively:

$$\begin{gathered}\Big(\Big\{(0{\perp\!\!\!\perp}1 \mid 2)\Big\},(0{\perp\!\!\!\perp}1 \mid 2)\Big)\\ \Big(\Big\{(0{\not\!\perp\!\!\!\perp}3 \mid 2)\Big\},(0{\not\!\perp\!\!\!\perp}3 \mid 2)\Big)\\ \Big(\Big\{(0{\perp\!\!\!\perp}3 \mid \{1,2\})\Big\},(0{\perp\!\!\!\perp}3 \mid \{1,2\})\Big) \qquad (17)\end{gathered}$$

while rule (16) corresponds to the argument

$$\Big(\Big\{(0{\perp\!\!\!\perp}1 \mid 2),(0{\not\!\perp\!\!\!\perp}3 \mid 2)\Big\},(0{\not\!\perp\!\!\!\perp}3 \mid \{1,2\})\Big). \qquad (18)$$

Let us extend this IKB with the following preference values for its propositions and rule.

$$\begin{aligned} Pref[(0{\perp\!\!\!\perp}1 \mid 2)] &= 0.8\\ Pref[(0{\not\!\perp\!\!\!\perp}3 \mid 2)] &= 0.7\\ Pref[(0{\perp\!\!\!\perp}3 \mid \{1,2\})] &= 0.5.\end{aligned}$$

According to Definition (12), the preference of each argument $(\{\sigma\},\sigma)$ *is equal to the preference value of* $\{\sigma\}$ *which is equal to the preference of* σ*, as it contains only a single proposition. Thus,*

$$\begin{aligned} Pref\Big[\Big(\Big\{(0{\perp\!\!\!\perp}1 \mid 2)\Big\},(0{\perp\!\!\!\perp}1 \mid 2)\Big)\Big] &= 0.8\\ Pref\Big[\Big(\Big\{(0{\not\!\perp\!\!\!\perp}3 \mid 2)\Big\},(0{\not\!\perp\!\!\!\perp}3 \mid 2)\Big)\Big] &= 0.7\\ Pref\Big[\Big(\Big\{(0{\perp\!\!\!\perp}3 \mid \{1,2\})\Big\},(0{\perp\!\!\!\perp}3 \mid \{1,2\})\Big)\Big] &= 0.5.\end{aligned}$$

The preference of argument (18) equals the preference of the set of its antecedents, which, according to Eq. (12), is equal to the product of their individual preferences, that is,

$$Pref\Big[\Big(\Big\{(0{\perp\!\!\!\perp}1 \mid 2),(0{\not\!\perp\!\!\!\perp}3 \mid 2)\Big\},(0{\not\!\perp\!\!\!\perp}3 \mid \{1,2\})\Big)\Big] = 0.8\times 0.7 = 0.56.$$

Proposition (15) and rule (16) contradict each other logically, that is, their corresponding arguments (17) and (18) defeat each other. However, argument (18) is not attacked as its preference is 0.56 *which is larger than* 0.5*, the preference of its defeater argument (17). Since no other argument defeats (18), it is acceptable, and (17), being attacked by an acceptable argument, must be rejected. We therefore see that using preferences the inconsistency of Example 1 has been resolved in favor of rule (16).*

Let us now illustrate the defend relation, that is, how an argument can be defended by some other argument. The example also illustrates an alternative resolution for the inconsistency of Example 1, this time in favor of the independence proposition (15).

Example 4. *Let us extend the IKB of Example 3 with two additional independence propositions and an additional rule. The new propositions and their preference are:*

$$\begin{aligned} Pref[(0{\perp\!\!\!\perp}4 \mid \{2,3\})] &= 0.9\\ Pref[(0{\perp\!\!\!\perp}3 \mid \{2,4\})] &= 0.8\end{aligned}$$

and the new rule is:

$$(0 \perp\!\!\!\perp 4 \mid \{2,3\}) \wedge (0 \perp\!\!\!\perp 3 \mid \{2,4\}) \quad \Longrightarrow \quad (0 \perp\!\!\!\perp 3 \mid 2).$$

This rule is an instance of the Intersection axiom followed by Decomposition.

The corresponding arguments and preferences are:

$$\begin{aligned} Pref\left[\left(\left\{(0 \perp\!\!\!\perp 4 \mid \{2,3\})\right\}, (0 \perp\!\!\!\perp 4 \mid \{2,3\})\right)\right] &= 0.9 \\ Pref\left[\left(\left\{(0 \perp\!\!\!\perp 3 \mid \{2,4\})\right\}, (0 \perp\!\!\!\perp 3 \mid \{2,4\})\right)\right] &= 0.8 \end{aligned}$$

corresponding to the two propositions, and

$$Pref\left[\left(\left\{(0 \perp\!\!\!\perp 4 \mid \{2,3\}), (0 \perp\!\!\!\perp 3 \mid \{2,4\})\right\}, (0 \perp\!\!\!\perp 3 \mid 2)\right)\right] = 0.9 \times 0.8 = 0.72 \qquad (19)$$

corresponding to the rule.

As in Example 3, argument (17) is attacked by argument (18). Let us represent this graphically using an arrow from argument a to argument b to denote that a attacks b, that is,

$$Argument\ (18) \longrightarrow Argument\ (17).$$

If the IKB was as in Example 3, (18) would had been accepted and (17) would have been rejected. However, the additional argument (19) now defeats (undercuts) (18) by logically contradicting its antecedent $(0 \not\perp\!\!\!\perp 3 \mid 2)$. *Since the preference of (19), namely* 0.72, *is larger than that of (18), namely* 0.56, *(19) attacks (18). Therefore, (19) defends all arguments that are attacked by argument (18), and in particular (17). Graphically,*

$$Argument\ (19) \longrightarrow Argument\ (18) \longrightarrow Argument\ (17).$$

Note this is not sufficient for accepting (17) as it has not been proved that its defender (19) is itself acceptable. We leave the proof of this as an exercise for the reader.

4. The Argumentative Independence Test (AIT)

The independence-based preference argumentation framework described in the previous section provides a semantics for the acceptance of arguments consisting of independence propositions. However, what we need is a procedure for a test of independence that, given as input a triplet $\sigma = (X,Y \mid \mathbf{Z})$ responds whether X is independent or dependent of Y given $\mathbf{Z}$. In other words, we need a semantics for the acceptance of propositions, not arguments. Let us consider the two propositions related to the input triplet $\sigma = (X,Y \mid \mathbf{Z})$, proposition $(\sigma = \texttt{true})$, abbreviated $\sigma_{\texttt{t}}$, and proposition $(\sigma = \texttt{false})$, abbreviated $\sigma_{\texttt{f}}$, that correspond to independence $(X \perp\!\!\!\perp Y \mid \mathbf{Z})$ and dependence $(X \not\perp\!\!\!\perp Y \mid \mathbf{Z})$ of σ, respectively. The basic idea for deciding on the independence or dependence of input triplet σ is to define a semantics for the acceptance or rejection of propositions $\sigma_{\texttt{t}}$ and $\sigma_{\texttt{f}}$ based on the acceptance or rejection of their respective propositional arguments $(\{\sigma_{\texttt{t}}\},\sigma_{\texttt{t}})$ and $(\{\sigma_{\texttt{f}}\},\sigma_{\texttt{f}})$. Formally,

$$\begin{aligned} (X \not\perp\!\!\!\perp Y \mid \mathbf{Z}) \text{ is accepted} \quad &\text{iff} \quad (\{(X \not\perp\!\!\!\perp Y \mid \mathbf{Z})\}, (X \not\perp\!\!\!\perp Y \mid \mathbf{Z})) \text{ is accepted, and} \\ (X \perp\!\!\!\perp Y \mid \mathbf{Z}) \text{ is accepted} \quad &\text{iff} \quad (\{(X \perp\!\!\!\perp Y \mid \mathbf{Z})\}, (X \perp\!\!\!\perp Y \mid \mathbf{Z})) \text{ is accepted.} \end{aligned} \qquad (20)$$

Based on this semantics over propositions, we decide on the dependence or independence of triplet σ as follows:

$$\begin{aligned}\sigma_{\mathrm{t}} = (X \perp\!\!\!\perp Y \mid \mathbf{Z}) \text{ is accepted} \quad &\Longrightarrow \quad (X \perp\!\!\!\perp Y \mid \mathbf{Z}) \\ \sigma_{\mathrm{f}} = (X \not\!\perp\!\!\!\perp Y \mid \mathbf{Z}) \text{ is accepted} \quad &\Longrightarrow \quad (X \not\!\perp\!\!\!\perp Y \mid \mathbf{Z}). \end{aligned} \tag{21}$$

We call the test that determines independence in this manner the **Argumentative Independence Test** or **AIT**. For the above semantics to be well-defined, a triplet σ must be either independent or dependent, that is, not both or neither. For that, exactly one of the antecedents of the above implications of Eq. (21) must be true. Formally,

Theorem 18. *For any input triplet $\sigma = (X, Y \mid \mathbf{Z})$, the argumentative independence test (**AIT**) defined by Eqs. (20) and (21) produces a non-ambiguous decision, that is, it decides σ evaluates to either independence or dependence, but nor both or neither.*

For that to happen, one and only one of its corresponding propositions σ_{t} or σ_{f} must be accepted. A necessary condition for this is given by the following theorem.

Theorem 19. *Given a PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ with a strict and transitive preference relation π, every propositional argument $(\{\sigma_t\}, \sigma_t) \in \mathcal{A}$ and its negation $(\{\sigma_f\}, \sigma_f)$ satisfy*

$$(\{\sigma_t\}, \sigma_t) \textit{ is accepted iff } (\{\sigma_f\}, \sigma_f) \textit{ is rejected.}$$

The above theorem is not sufficient because the propositions may still be in abeyance, but this possibility is ruled out for strict preference relations by Theorem 20, presented in the next section.

The formal proofs of Theorems 18, 19 and 20 are presented in Appendix B. We now illustrate the use of AIT with an example.

Example 5. *We consider an extension of Example 3 to illustrate the use of the AIT to decide on the independence or dependence of input triplet $(0, 3 \mid \{1, 2\})$. According to Eq. (20) the decision depends on the status of the two propositional arguments:*

$$(\{(0 \not\!\perp\!\!\!\perp 3 \mid \{1,2\})\}, (0 \not\!\perp\!\!\!\perp 3 \mid \{1,2\})), \textit{ and} \tag{22}$$
$$(\{(0 \perp\!\!\!\perp 3 \mid \{1,2\})\}, (0 \perp\!\!\!\perp 3 \mid \{1,2\})). \tag{23}$$

Argument (23) is equal to argument (17) of Example 3 that was proved to be rejected in that example. Therefore, according to Theorem 19, its negated propositional argument Eq. (22) must be accepted, and we can conclude that triplet $(0, 3 \mid \{1, 2\})$ corresponds to a dependence, that is, we conclude that $(0 \not\!\perp\!\!\!\perp 3 \mid \{1, 2\})$.

5. The Top-down AIT Algorithm

We now discuss in more detail the top-down algorithm which is used to implement the argumentative independence test, introduced in Section 3. We start by simplifying the recursion of Eq. (7) that determines the state (accepted, rejected, or in abeyance) of an argument *a*. We then explain the algorithm and analyze its computability (i.e., prove that its recursive execution is always finite) and its time complexity.

To simplify the recursion Eq. (7) we use the following theorem (proved in Appendix B).

Theorem 20. *Let $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ be a PAF with a strict preference relation π. Then no argument $a \in \mathcal{A}$ is in abeyance.*

This theorem reduces the number of states of each argument to two, that is, an argument can be either accepted or not accepted (rejected). We will use the name of the argument a to denote the predicate "a is accepted" and its negation $\neg a$ to denote the predicate "a is rejected." With this notation, the above theorem, and the fact that we have extended the semantics of acceptability from the defeat to the attack relation (using preferences), the recursion of Eq. (7) can be expressed as follows

$$\begin{aligned} a &\iff \forall b \in attackers(a),\ \neg b \\ \neg a &\iff \exists b \in attackers(a),\ b \end{aligned}$$

or, equivalently,

$$\begin{aligned} a &\iff \bigwedge_{b \in attackers(a)} \neg b \\ \neg a &\iff \bigvee_{b \in attackers(a)} b. \end{aligned}$$

Finally, we notice that the second formula is logically equivalent to the first (simply negating both sides of the double implication recovers the first). Therefore, the Boolean value of the dialog tree for a can be computed by the simple expression

$$a \iff \bigwedge_{b \in attackers(a)} \neg b. \tag{24}$$

To illustrate, consider an attacker b of a. If b is rejected, that is, $\neg b$, the conjunction on the right cannot be determined without examining the other attackers of a. Only when all attackers of a are known to be rejected can the value of a be determined, that is, accepted. Instead, if b is accepted, that is, b, the state of $\neg b$ is `false` and the conjunction can be immediately evaluated to `false`, that is, a is rejected regardless of the acceptability of any other attackers.

An iterative version of the top-down algorithm is shown in Algorithm 3. We assume that the algorithm can access a global PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$, with arguments in $\mathcal{A}$ defined over a knowledge base $\mathcal{K} = \langle \Sigma, \Psi \rangle$. Given as input a triplet $t = (X, Y \mid \mathbf{Z})$, if the algorithm returns `true` (`false`) then we conclude that t is independent (dependent). It starts by creating a root node u for the propositional argument U of proposition $t =$ `true` (lines 1–6). According to Eqs. (20) and (21), the algorithm then decides `true` if U is accepted (line 22). Otherwise, the algorithm returns `false` (line 23). This is because in this case, according to Theorem 19, the negation of propositional argument U must be accepted.

Algorithm 3 is an iterative version of a tree traversal algorithm. It maintains a queue of the nodes that have not been expanded yet. A node is expanded when its children are added to the tree. In the algorithm, this is done in the loop of lines 17 to 21, which uses subroutine *getAttackers* of Algorithm 5 to obtain all attackers of an argument. This subroutine finds all attackers of the input argument a in a backward-chaining fashion, that is, given an argument $a = (H, h)$, it searches for all rules in the knowledge base $\mathcal{K}$ whose consequent matches the negation of some proposition in the

Algorithm 3 *independent*(*triplet t*).

```
1:  $f_{\texttt{true}} \leftarrow$ proposition $(t = \texttt{true})$    /* Creates independence proposition $(t = \texttt{true})$. */
2:  $U_{\texttt{true}} \leftarrow (\{f_{\texttt{true}}\}, f_{\texttt{true}})$
3:  $u_{\texttt{true}} \leftarrow$ node for argument $U_{\texttt{true}}$
4:  $u_{\texttt{true}}.parent \leftarrow \texttt{nil}$
5:  $u.STATE \leftarrow \texttt{nil}$
6:  $fringe \leftarrow [u]$    /* Initialize with u (root). */
7:  /* Create global rejected node, denoted by $\rho$. */
8:  $\rho \leftarrow$ node with no argument and state rejected
9:  while $fringe \neq \varnothing$ do
10:     $u \leftarrow dequeue(fringe)$
11:     $attackers \leftarrow getAttackers(u.argument)$
12:     if $(attackers = \varnothing)$ then
13:         $u.STATE \leftarrow \texttt{accepted}$
14:         if $sendMsg(\rho, u) = terminate$ then break
15:     $attackers \leftarrow$ sort attackers in decreasing order of preference.
16:     /* Enqueue attackers after decomposing them. */
17:     for each $A \in attackers$ do
18:         $a \leftarrow$ node for argument $A$
19:         $a.parent \leftarrow u$
20:         $a.STATE \leftarrow \texttt{nil}$
21:         enqueue $a$ in $fringe$    /* See details in text. */
22: if $(u.STATE = \texttt{accepted})$ then return true
23: if $(u.STATE = \texttt{rejected})$ then return false
```

Algorithm 4 *sendMsg*(*Node c*, *Node p*).

```
1:  /* Try to evaluate node p given new information in c.STATE */
2:  if $p \neq \texttt{nil}$ then
3:      if $c.STATE = \texttt{accepted}$ then $p.STATE \leftarrow \texttt{rejected}$
4:      else if ($\forall$ children $q$ of $p$, $q.STATE \neq \texttt{rejected}$) then $p.STATE \leftarrow \texttt{accepted}$
5:      /* If p was successfully evaluated, try to evaluate its parent by sending message upward. */
6:      if $p.STATE \neq \texttt{nil}$ then
7:          return $sendMsg(p, p.parent)$
8:      else
9:          return continue
10: else
11:     return terminate /* The root node has been evaluated. */
```

support H (undercutters), or the negation of its head h (rebutters). Every node maintains a three-valued state variable *STATE* $\in \{\texttt{nil}, \texttt{accepted}, \texttt{rejected}\}$. The `nil` state denotes that the value of the node is not yet known, and a node is initialized to this state when it is added to the tree.

The algorithm recurses down the dialog tree until a node is found that has no attackers (line 12). Such a node is accepted in line 13, that is, the conjunction of Eq. (24) is trivially true, and its *STATE* is propagated upwards toward the root to the parent using subroutine *sendMsg* (Algorithm 4). Every

Algorithm 5 Finds all attackers of input argument a in knowledge base $\mathcal{K} = \langle \Sigma, \Psi \rangle$: *getAttackers*$(a = (H, h))$

```
1: $attackers \leftarrow \varnothing$
2: /* Get all undercutters or rebutters of a. */
3: for all propositions $\varphi \in H \cup \{h\}$ do
4:     /* Get all defeaters of proposition $\varphi$. */
5:     for all rules $(\Phi_1 \wedge \Phi_2 \ldots \wedge \Phi_n \implies \neg\varphi) \in \Psi$ do
6:         /* Find all propositionalizations of the rule whose consequent matches $\neg\varphi$. */
7:         for all subsets $\{\varphi_1, \varphi_2 \ldots, \varphi_n\}$ of $\Sigma$ s.t. $\Phi_1 \equiv \varphi_1, \Phi_2 \equiv \varphi_2 \ldots \Phi_n \equiv \varphi_n$ do
8:             $d \leftarrow (\{\varphi_1 \wedge \varphi_2 \ldots \varphi_n\}, \neg\varphi)$ /* Create defeater. */
9:             /* Is the defeater an attacker? */
10:            if $\neg(a \gg_\pi d)$ then
11:                $attackers \leftarrow attackers \cup \{d\}$
12: return $attackers$
```

time a node receives a message from a child, if the message is `accepted`, the node is rejected (line 3 of Algorithm 4), otherwise the node is accepted if all its children has been evaluated to `rejected` (line 4 of Algorithm 4). The subroutine *sendMsg* then proceeds recursively by forwarding a message to the parent whenever a node has been evaluated (line 7). If the root is reached and evaluated, the message is sent to its parent, which is `nil`. In this case, the subroutine returns the special keyword *terminate* back to the caller, indicating that the root has been evaluated and thus the main algorithm (Algorithm 3) can terminate. The caller can be either the subroutine *sendMsg*, in which case it pushes the returned message up the method-calling stack, or the top-down algorithm in line 14, in which case its "while" loop is terminated.

An important part of the algorithm is yet underspecified, namely the order in which the attackers of a node are explored in the tree (i.e., the priority with which nodes are enqueued in line 21). This affects the order of expansion of the nodes in the dialog tree. Possible orderings are depth-first, breadth-first, iterative deepening, as well as informed searches such as best-first when a heuristic is available. In our experiments we used iterative deepening because it combines the benefits of depth-first and breadth-first search, that is, small memory requirements on the same order as depth-first search (i.e., on the order of the maximum number of children a node can have) but also the advantage of finding the shallowest solution like breadth-first search. We also used a heuristic for enqueuing the children of a node. According to iterative deepening, the position in the queue of the children of a node is specified relative to other nodes, but not relative to each other. We therefore specified the relative order of the children according to the value of the preference function: children with higher preference are enqueued first (line 15 of the top-down algorithm), and thus, according to iterative deepening, would be dequeued first.

5.1 Computability of the Top-Down Algorithm

An open question is whether the top-down algorithm is computable, that is, whether it always terminates. In this section we prove that it is. To prove this we need to show that under certain general conditions the acceptability of an argument a can always be determined, that is, that the algorithm always terminates. This is proved by the following theorem.

Theorem 21. *Given an arbitrary triplet $t = (X,Y \mid \mathbf{Z})$, and a PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ with a strict preference relation π, Algorithm 3 with input t over $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ terminates.*

The proof consists on showing that the path from the root a to any leaf is always finite. For that, the concept of an attack sequence is needed.

Definition 22. *An* attack sequence *is a sequence $\langle a_1, a_2, \ldots, a_n \rangle$ of n arguments such that for every $2 \leq i \leq n$, a_i attacks a_{i-1}.*

By the manner in which the top-down algorithm constructs the dialog tree it is clear that any path from the root to a leaf is an attack sequence. It therefore suffices to show that any such sequence is finite. This is done by the following theorem.

Theorem 23. *Every attack sequence $\langle a_1, a_2, \ldots, a_n \rangle$ in a PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ with strict π and finite $\mathcal{A}$ is finite.*

Intuitively, if the preference relation is strict then an element can attack its predecessor in the sequence but not vice versa. Since the set of arguments $\mathcal{A}$ is finite, the only way for an attack sequence to be infinite is to contain a cycle. In that case, an argument would be attacking at least one of its predecessors, which cannot happen in a PAF with a strict preference relation. We present formal proofs of Theorems 21 and 23 in Appendix A.

We thus arrived at the important conclusion that, under a strict preference function and a finite argument set, the state of any argument is computable. As we showed in Section 3.3, the preference function for independence knowledge bases is strict, and thus the computability of the top-down algorithm is guaranteed.

5.2 Computational Complexity of the Top-Down Algorithm

Since Algorithm 3 is a tree traversal algorithm, its time complexity can be obtained by techniques contained in standard algorithmic texts, for example, Cormen et al. (2001). The actual performance depends on the tree exploration procedure. In our case we used iterative deepening due to its linear memory requirements in d, where d is the smallest depth at which the algorithm terminates. Iterative deepening has a worst-time time complexity of $O(b^d)$, where b is an upper bound on the *dialog tree branching factor*. Therefore, for a constant $b > 1$ the execution time is exponential in d in the worst case. Furthermore, for the case of independence tests, b itself may also be exponential in n (the number of variables in the domain). This is because the inference rules of Eqs. (5) and (6) are universally quantified, and therefore their propositionalization (lines 7–11 of Algorithm 5), may result in an exponential number of rules with the same consequent (attackers). A tighter bound may be possible but, lacking such a bound, we introduce in the next section an approximate top-down algorithm, which reduces the running time to polynomial. As we show in our experiments, the use of this approximation does not appreciably affect the accuracy improvement due to argumentation.

6. The Approximate Top-Down AIT Algorithm

As the top-down algorithm has a theoretically exponential running time in the worst case, we hereby present a practical polynomial-time approximation of the top-down algorithm. We make use of two approximations: (a) To address the exponential behavior due to the depth of the dialog tree we apply a *cutoff depth d* for the process of iterative deepening. (b) To address the potentially exponential size of the branching factor b (which equals the maximum number of defeaters of any argument appearing in the dialog tree) we limit the number of defeaters of each node—thus bounding the number of its attackers/children—to a polynomial function of n (the domain size) during the propositionalization process of Algorithm 5 (lines 7–11). Let (H,h) be an argument and let $\varphi \in H \cup \{h\}$ be one of its propositions, as in line 3 of Algorithm 5. The set of attackers Σ_φ of (H,h) consists of all rules $\{\varphi_1 \wedge \varphi_2 \ldots \wedge \varphi_k \implies \neg\varphi\}$ of Σ, for some constant upper bound k on the size of their support. If $\varphi = (\mathbf{X}, \mathbf{Y} \mid \mathbf{Z})$ and $\varphi_i = (\mathbf{X}_i, \mathbf{Y}_i \mid \mathbf{Z}_i)$ for all $1 \leq i \leq k$, then our approximation generates and uses a subset of Σ_φ in the dialog tree such that

$$\begin{array}{rcccl} |\mathbf{X}| - c & \leq & |\mathbf{X}_i| & \leq & |\mathbf{X}| + c \\ |\mathbf{Y}| - c & \leq & |\mathbf{Y}_i| & \leq & |\mathbf{Y}| + c \\ |\mathbf{Z}| - c & \leq & |\mathbf{Z}_i| & \leq & |\mathbf{Z}| + c \end{array} \tag{25}$$

where $|\cdot|$ denotes set cardinality, and c is a user-specified integer parameter that defines the approximation. We call this the **approximate top-down algorithm**. The computational complexity of the approximate top-down algorithm is polynomial in n, as shown in the next section.

6.1 Test Complexity of the Approximate Top-Down Algorithm

In this section we prove that the number of statistical tests required by the Approximate Top-Down algorithm is polynomial in n. As described in the previous section, the approximate algorithm generates a bounded number of attackers for each proposition in the argument corresponding to some node in the dialog tree. A bound on the number of the possible attackers can be defined by the approximation of Eq. (25). These equations dictate that the size of each possible set $\mathbf{X}_i$ in some proposition $(\mathbf{X}_i, \mathbf{Y}_i \mid \mathbf{Z}_i)$ of some attacker of proposition $(\mathbf{X}, \mathbf{Y} \mid \mathbf{Z})$ is between $|\mathbf{X}| + c$ and $|\mathbf{X}| - c$ (inclusively). As the number of elements that can be members of $\mathbf{X}_i$ is bounded by n (the domain size), this produces at most n^{2c+1} possible instantiations for set $\mathbf{X}_i$. Similarly, the number of possible instantiations for $\mathbf{Y}_i$ and $\mathbf{Z}_i$ is also n^{2c+1}. Therefore, an upper bound for the number of matches to some proposition in the antecedent of an attacking rule is $O(n^{6c+3})$ for some constant c. As there are r rules in the rule set and up to k propositions in each rule for some constants r and k (for example, $r = 5$ and $k = 3$ for Eq. (5) and $r = 8$ and $k = 4$ for Eq. (6)), an upper bound on the number of children of a node in the dialog tree is $O(rkn^{6c+3})$, and thus an upper bound on the number of nodes in the dialog tree of depth d is $O((rk)^d n^{d(6c+3)})$. As we demonstrate in our experiments, this is a rather loose upper bound and the performance of the approximate top-down algorithm is reasonable in practice, but it does serve to show that the theoretical worst-case performance is polynomial in n. In the experiments shown in the next section we used $c = 1$ and $d = 3$.

7. Experimental Results

We conducted experiments on sampled and real-world data sets for the purpose of (a) evaluating the accuracy improvement of the argumentative test (both the exact and approximate versions) over its statistical counterpart; (b) demonstrating the performance improvements that can be achieved by the approximate version compared to the exact counterpart, without significant reduction in accuracy improvement; and (c) evaluating the improvements that result by the use of the argumentative framework for causal discovery. We address these issues below.

7.1 Comparative Evaluation of Bottom-Up, Exact Top-Down, and Approximate Top-Down Argumentative Tests

In this section we demonstrate that the argumentation approach, implemented either by the (exact) bottom-up or the exact top-down algorithm (Algorithm 3), improves the accuracy of independence tests on small data sets. We also show that the approximate top-down algorithm (see Section 6) has accuracy performance improvements similar to its exact counterpart but significantly better execution times (orders of magnitude), that make it more practical and usable for larger domains. As the output of the bottom-up algorithm is guaranteed to be equal to the exact top-down algorithm as Theorem 9 of Section 3, we omit accuracy results for the bottom-up algorithm here.

As the exact algorithm is impractical for large domains, for the present comparison we sampled data sets from two randomly generated Bayesian networks with $n = 8$ nodes. The networks were generated using *BNGenerator* (Ide et al., 2002), a publicly available Java package, with maximum degree per node τ equal to 3 and 7 to evaluate the performance in sparsely as well as densely connected domains. A large data set D was sampled from each network and our experiments were conducted on subsets of it containing an increasing number of data points N. This was done in order to assess the accuracy on varying conditions of reliability, as the reliability of a test varies (typically increases) with the amount of data available. To reduce variance, each experiment was repeated for ten data subsets of equal size, obtained by permuting the data points of D randomly and using the first N of them as input to our algorithms.

We first compare the accuracy of argumentative tests versus their purely statistical counterparts (i.e., the G^2 test) on several data sets sampled from randomly generated Bayesian networks. Sampled data experiments have the advantage of a more precise estimation of the accuracy since the underlying model is known. We present experiments for two versions of the exact top-down argumentative test, one using Pearl's general axioms of of Eq. (5), denoted $\mathbf{AIT_t\text{-}G}$, and another that uses Pearl's "directed" axioms of Eq. (6), denoted $\mathbf{AIT_t\text{-}D}$, as well as two versions of the approximate top-down argumentative test, denoted $\widehat{\mathbf{AIT}}_\mathbf{t}\mathbf{\text{-}G}$ and $\widehat{\mathbf{AIT}}_\mathbf{t}\mathbf{\text{-}D}$ respectively. We abbreviate purely statistical independence tests as **SIT**.

We report the estimated accuracy, which, for each data set, is calculated by comparing the result of a number of conditional independence tests (SITs or AITs) on data with the true value of independence, computed by querying the underlying model for the conditional independence of the same test using d-separation. Since the number of possible tests is exponential, we estimated the independence accuracy by randomly sampling a set $\mathcal{T}$ of 1,000 triplets $(X, Y, \mathbf{Z})$, evenly distributed among all possible conditioning set sizes $m \in \{0, \ldots, n-2\}$, that is, $1000/(n-1)$ tests for each m. The independence or dependence value of the triplets (in the true, underlying model) were not controlled, but left to be decided randomly. This resulted in a non-uniform distribution of dependencies and independences. For instance, in the experiments shown next ($n = 8$, $\tau = 3, 7$), the average proportion of independences vs. dependencies was 36.6% to 63.4% respectively for $\tau = 3$, and 11.4% to 88.6% respectively for $\tau = 7$. Denoting a triplet in $\mathcal{T}$ by t, by $I_{\text{true}}(t)$ the result of a test on t performed on the underlying model, and by $I_{\text{data-}\mathcal{Y}}(t)$ the results of performing a test on t of type $\mathcal{Y}$ on data, for $\mathcal{Y}$ equal to SIT, AIT_t-G, AIT_t-D, $\widehat{\text{AIT}}_\text{t}$-G, or $\widehat{\text{AIT}}_\text{t}$-D, the estimated accuracy of test type $\mathcal{Y}$ is defined as

$$\widehat{acc}^{\text{data}}_{\mathcal{Y}} = \frac{1}{|\mathcal{T}|}\left|\left\{t \in \mathcal{T} \mid I_{\text{data-}\mathcal{Y}}(t) = I_{\text{true}}(t)\right\}\right|.$$

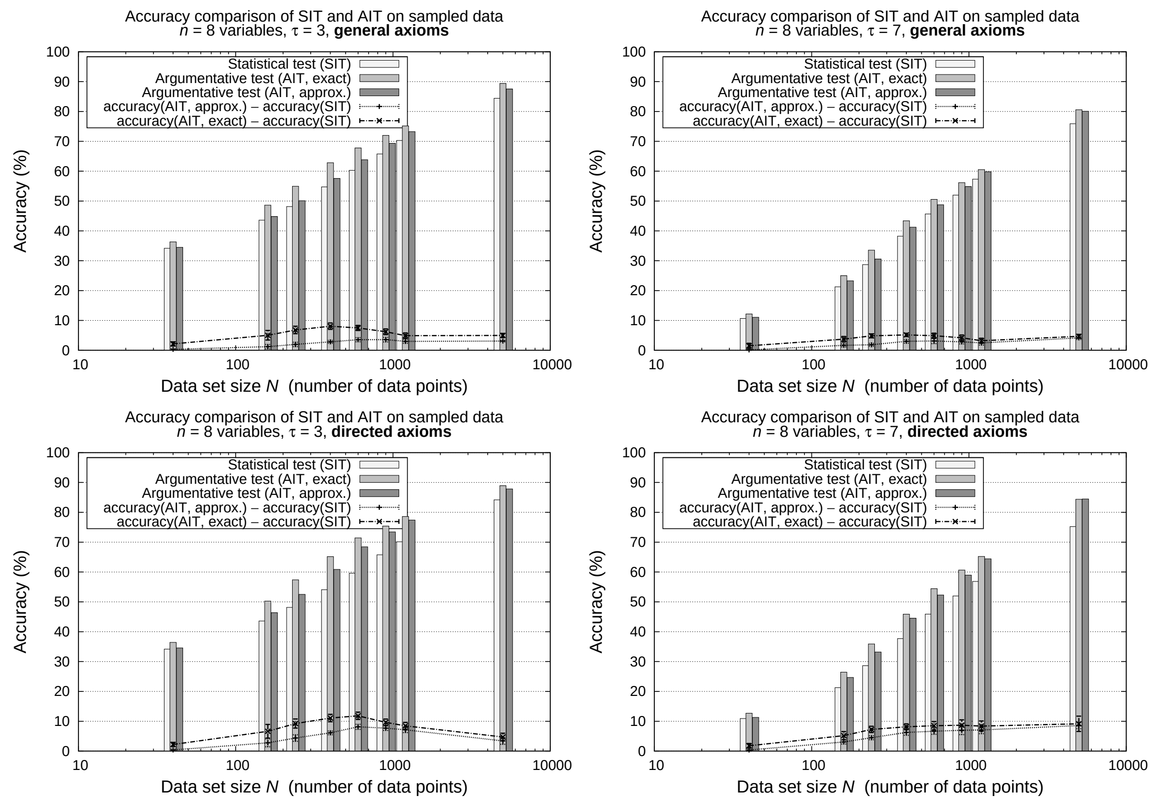

Figure 2: Accuracy comparison of statistical tests (SIT) vs. exact and approximate argumentative tests for domain size $n = 8$ and maximum degree per node $\tau = 3, 7$. The histograms show the absolute value of the accuracy while the line curves show the difference between SIT and the argumentative tests. 95% confidence intervals are also shown for the line graphs. **Top row:** General axioms. **Bottom row:** Directed axioms.

Figure 2 (top row) shows a comparison of the SIT with the exact and approximate top-down argumentative test over the general axioms for data set with increasing number of data points. The figure shows two plots for $\tau = 3, 7$ of the mean values (over runs for ten different data sets) of $\widehat{acc}^{\text{data}}_{\text{SIT}}$, $\widehat{acc}^{\text{data}}_{\text{AIT}_t\text{-G}}$, and $\widehat{acc}^{\text{data}}_{\widehat{\text{AIT}_t}\text{-G}}$ (histograms) and the difference between the accuracies of the AIT tests and the statistical one (line graphs) for various data set sizes N. A positive value of the difference corresponds to an improvement of the argumentative test over SIT. The plots also show the statistical significance of this difference with 95% confidence intervals (error bars), that is, the interval around the mean value that has a 0.95 probability of containing the true difference. The figure demonstrates that there exist modest but consistent and statistically significant improvements in the accuracy of both the exact and approximate argumentative tests over the statistical test. We can observe improvements over the entire range of data set sizes in both cases with maximum improvements of up to 9% and 6% for the exact and approximate cases respectively (at $\tau = 3$ and $N = 600$).

In certain situations where the experimenter knows that the underlying distribution belongs to the class of Bayesian networks, it is appropriate to use the specific axioms of Eq. (6) instead of the general axioms of Eq. (5). The bottom row of Figure 2 presents the same comparison as the top row but for the exact and approximate argumentative tests AIT_t-D and $\widehat{\text{AIT}}_t$-D that use the directed axioms instead of the general ones. As in the case for AIT using the general axioms, we can observe statistically significant improvements over the entire range of data set sizes in both cases. In this case however, the improvements are larger, with maximum increases in the accuracy of the exact and approximate test of up to 13% and 9% respectively (again for $\tau = 3$ and $N = 600$).

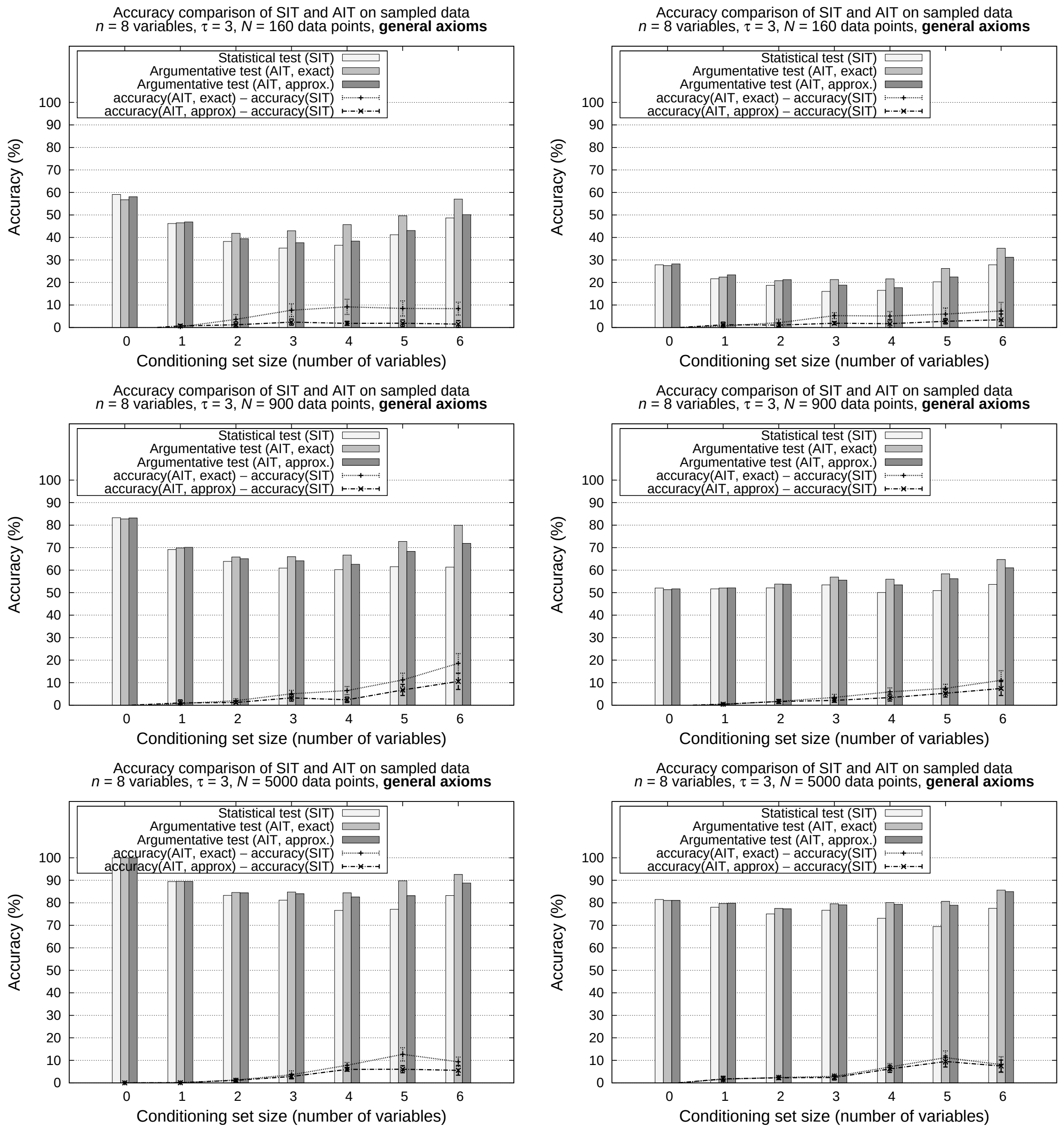

Figure 3: Accuracy comparison of SIT vs. exact (AIT_t-G) and approximate ($\widehat{\text{AIT}}_t$-G) argumentative tests over the general axioms for increasing conditioning set sizes. The six plots correspond to maximum degrees per node $\tau = 3, 7$, and data set sizes $N = 160, 900$ and 5000.

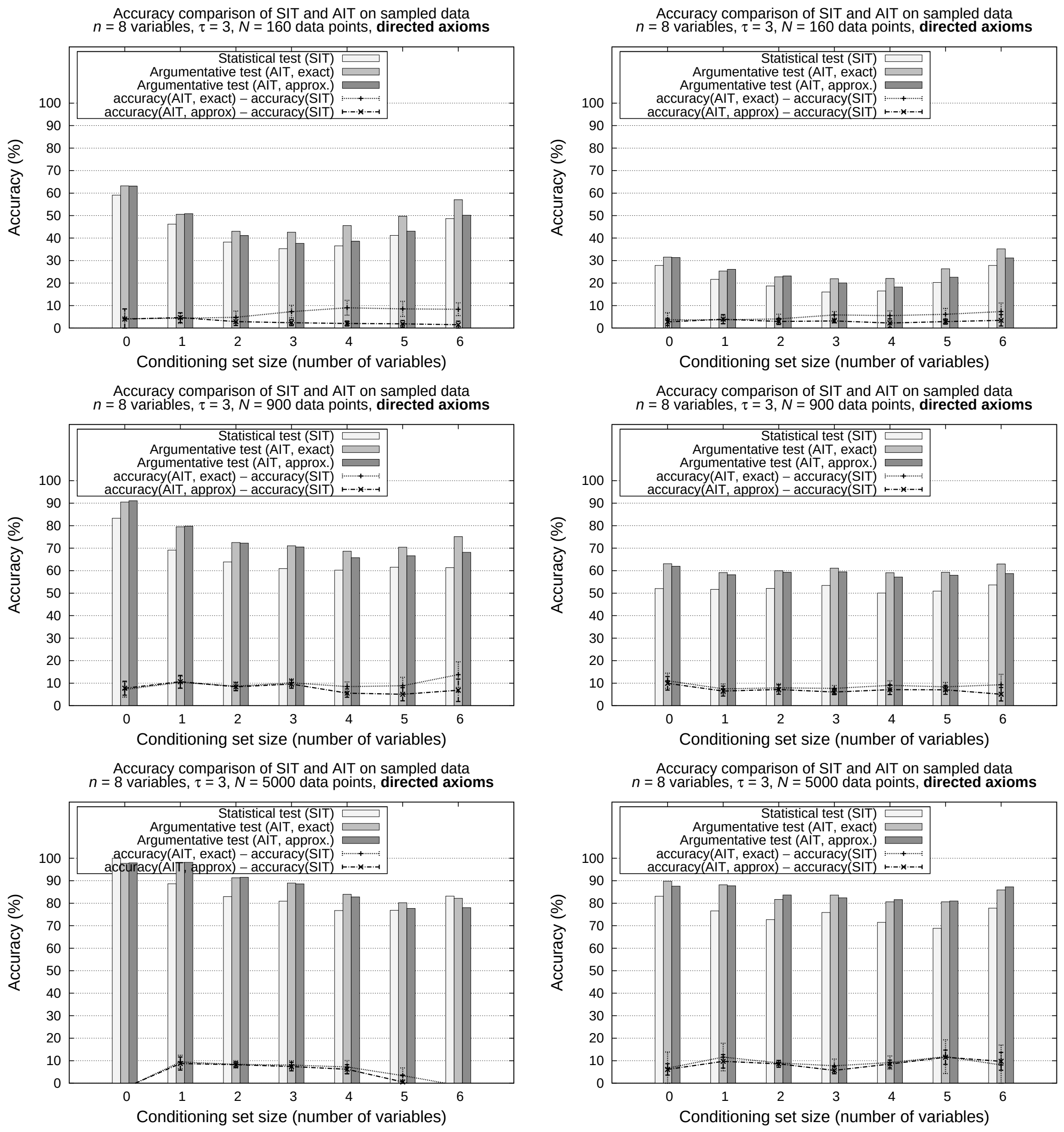

Figure 4: Same as Figure 3 but for AIT using the directed axioms instead of the general ones.

We also evaluated the accuracy of these tests for increasing conditioning set sizes. Figures 3 and 4 show a comparison of the SIT with the exact and approximate top-down argumentative test using the general and directed axioms respectively, for accuracies over increasing conditioning set size for data sizes $N = 160, 900$, and 5000 (different rows). We can observe statistically significant improvements over the entire range of conditioning set sizes in all twelve graphs. Except in one case (directed axioms, $N = 5000$, $\tau = 3$), all graphs show an upward trend in accuracy for increasing conditioning set size, representing a positive impact of the argumentative approach that increases with a decrease in test reliability, that is, increasing conditioning set size.

We also compared the execution times of the bottom-up, exact top-down and approximate top-down algorithms on the same data sets. To run the bottom-up algorithm we generated the set of all propositional arguments possible, that is, arguments of the form $(\{\sigma\}, \sigma)$, by iterating over all possible triplets $(\mathbf{X}, \mathbf{Y} \mid \mathbf{Z})$, and inserted them in the knowledge base together with their preference, as described in Section 3.1. Similarly, for the set of axioms that we used in each case, that is, either the general (Eq. (5)) or the specific ones (Eq. (6)), we iterated over all possible matches of each rule, inserting the corresponding (single-headed and decomposed) instantiated rule in the knowledge base again together with its preference. The reason for including all propositional and rule-based arguments in our IKB is to allow the argumentation framework to consider all possible arguments in favor of or against an independence query. We compared the bottom-up algorithm $\mathrm{AIT_b}$, the exact top-down algorithms $\mathrm{AIT_t}$, and the approximate top-down algorithm $\widehat{\mathrm{AIT}}_\mathrm{t}$. For this, we measured the time it takes to discover the structure of a Bayesian networks using three versions of the PC algorithm (Spirtes et al., 2000), each using one of the three argumentative tests $\mathrm{AIT_b}$, $\mathrm{AIT_t}$, or $\widehat{\mathrm{AIT}}_\mathrm{t}$ to conduct the independence tests. As usual, we consider two versions of each test $\mathrm{AIT_b}$, $\mathrm{AIT_t}$, and $\widehat{\mathrm{AIT}}_\mathrm{t}$, one that uses the general axioms of Eq. (5), that is, $\mathrm{AIT_b}$-G, $\mathrm{AIT_t}$-G, and $\widehat{\mathrm{AIT}}_\mathrm{t}$-G, respectively, and one that uses the specific axioms of Eq. (6) (applicable to Bayesian networks), that is, $\mathrm{AIT_b}$-D, $\mathrm{AIT_t}$-D, and $\widehat{\mathrm{AIT}}_\mathrm{t}$-D, respectively. The data sets used are the same as the ones used in the accuracy comparisons above.

Figure 5 plots the execution time of argumentative tests $\mathrm{AIT_b}$-G vs. $\mathrm{AIT_t}$-G vs. $\widehat{\mathrm{AIT}}_\mathrm{t}$-G (top row) and $\mathrm{AIT_b}$-D vs. $\mathrm{AIT_t}$-D vs. $\widehat{\mathrm{AIT}}_\mathrm{t}$-D (bottom row) for tests that were conducted by the PC algorithm while learning the structure. Note that both the *x* and *y*-axes are plotted in log-scale. We can observe improvements in the execution time of the exact top-down algorithm over that of the bottom-up algorithm of an order of magnitude over the entire range of data set sizes in all four plots. We can also see improvement of a similar order between the exact and approximate top-down argumentative algorithms. For instance, for the general axioms and $\tau = 3$ (top-left plot), the execution time for $N = 5000$ is 2749 seconds for the bottom-up against 107 seconds for the exact top-down and 15 seconds for the approximate top-down algorithm. We see even more pronounced execution time improvements when using the directed axioms (bottom row of Fig. 5).

The execution-time results demonstrate that the exact top-down algorithm performs significantly better than the bottom-up algorithm, while producing the exact same output (according to Theorem 9 of Section 3). This implies a clear advantage of using the top-down over the bottom-up algorithm. Furthermore, we also saw that the approximate top-down algorithm performs similarly in terms of accuracy improvement while having polynomial worst-case execution time and in practice being several orders of magnitude faster than the exact top-down algorithm, which is exponential in the worst-case. As in the next two sections we continue our evaluation on domains significantly larger than the $n = 8$ variables that we examined here, it would be difficult or impractical for the exact algorithms to be employed. For these reasons in the following experiments we use the more practical approximate algorithm, which can be applied to larger domains.

7.2 Causal Discovery in Larger Domains

We also conducted experiments that demonstrate the performance of the approximate top-down algorithm by (a) showing its applicability to large domains, and (b) demonstrating positive improvements in accuracy of argumentative tests on the learning of the structure of Bayesian networks, the main problem faced by causal discovery algorithms. In the following experiments we used the PC

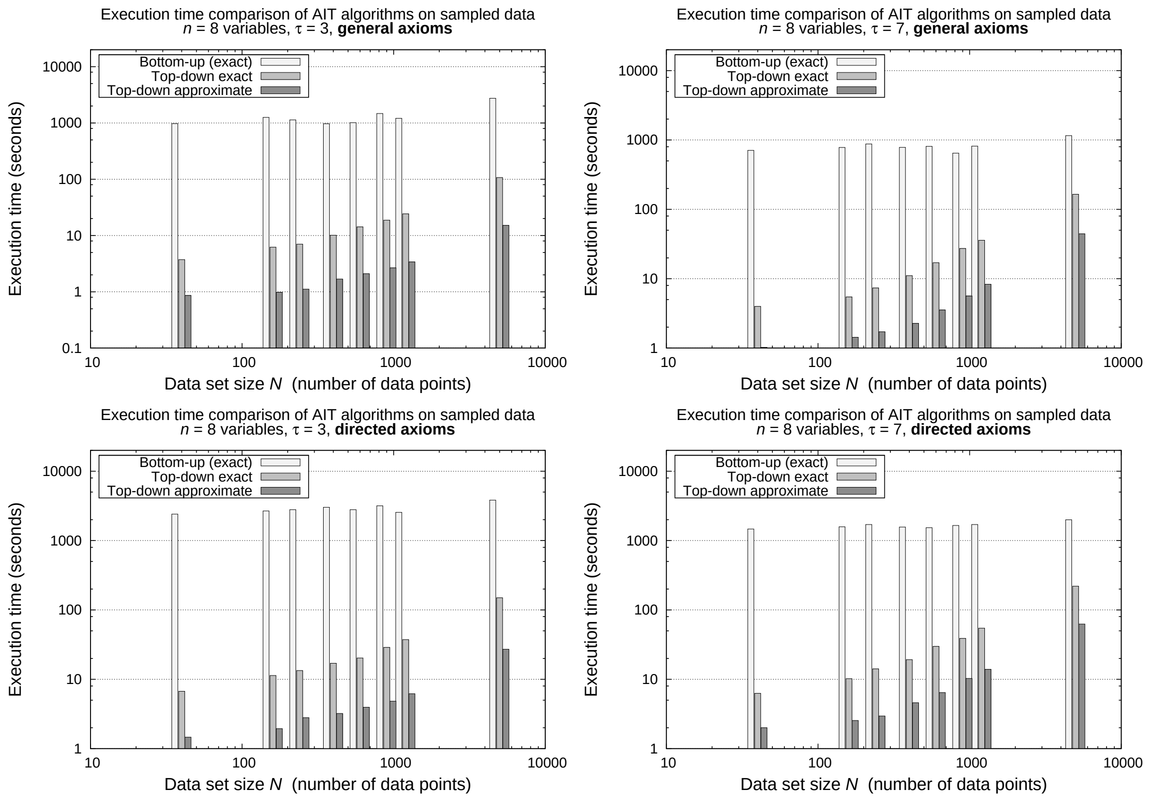

Figure 5: Execution time comparison for the PC algorithm when it uses the bottom-up and exact top-down and approximate top-down argumentative tests to learn the structure of a Bayesian network from data sampled from Bayesian models with domain size $n = 8$, maximum degrees $\tau = 3, 7$. The bars show the absolute value of the running time using a logarithmic scale. **Top row:** general axioms. **Bottom row:** directed axioms.

algorithm. We compared the true structure of the underlying model to the resulting structure of the PC algorithm when it uses SITs as independence tests, denoted PC-SIT, and its output when it uses argumentative independence tests, denoted PC-$\widehat{\mathrm{AIT}}_\mathrm{t}$-D, when using the directed axioms.

We evaluated the resulting networks by their ability to accurately represent the true independences in the domain, calculated by comparing the results (`true` or `false`) of a number of conditional tests conducted using d-separation on the output networks (PC-SIT or PC-$\widehat{\mathrm{AIT}}_\mathrm{t}$-D). Denoting by $\mathcal{T}$ this set of 2,000 triplets, by $t \in \mathcal{T}$ a triplet, by $I_\mathrm{true}(t)$ the result of a test performed on the underlying model, and by $I_{\mathrm{PC}\text{-}\mathcal{Y}}(t)$ the result of performing a d-separation test on the network output by the PC algorithm using the $\mathcal{Y}$ test, $\mathcal{Y}$ equal to SIT or $\widehat{\mathrm{AIT}}_\mathrm{t}$-D, the estimated accuracy is defined as

$$\widehat{acc}^{\mathrm{PC}}_{\mathcal{Y}} = \frac{1}{|\mathcal{T}|}\left|\left\{t \in \mathcal{T} \mid I_{\mathrm{PC}\text{-}\mathcal{Y}}(t) = I_\mathrm{true}(t)\right\}\right|. \tag{26}$$

We considered data sampled from randomly generated Bayesian networks of sizes $n = 24$, and maximum degrees $\tau = 3, 7$. For each network we sampled ten data sets, and, for each data set, we

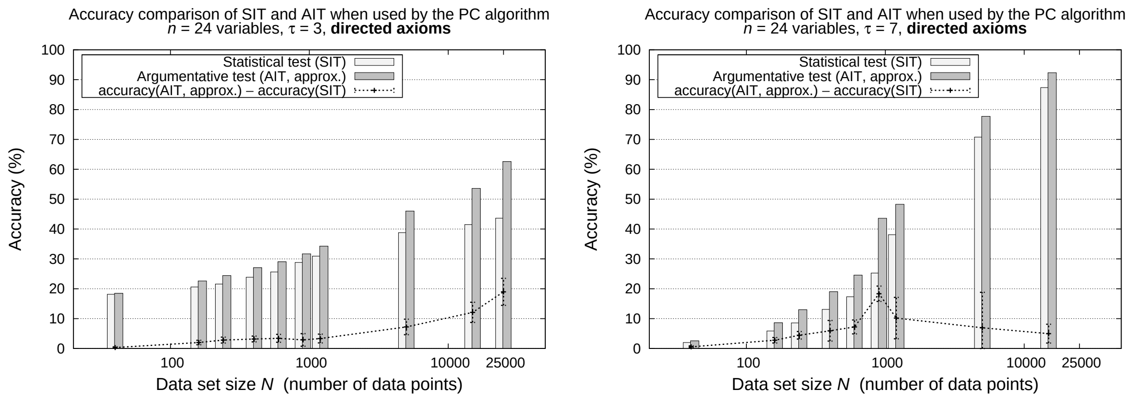

Figure 6: Comparison of statistical tests (SIT) vs. approximate argumentative tests on the directed axioms ($\widehat{\text{AIT}_t}$-D) for data sets sampled from Bayesian models for domain size $n = 24$ and maximum degrees $\tau = 3, 7$.

conducted experiments on subsets of D containing an increasing number of data points. We report the average over the ten data sets of the estimated accuracy calculated using Eq. (26), for $\mathcal{Y} = $ SIT or $\widehat{\text{AIT}_t}$-D, as well as the difference between the average accuracies including the 95% confidence interval for the difference.

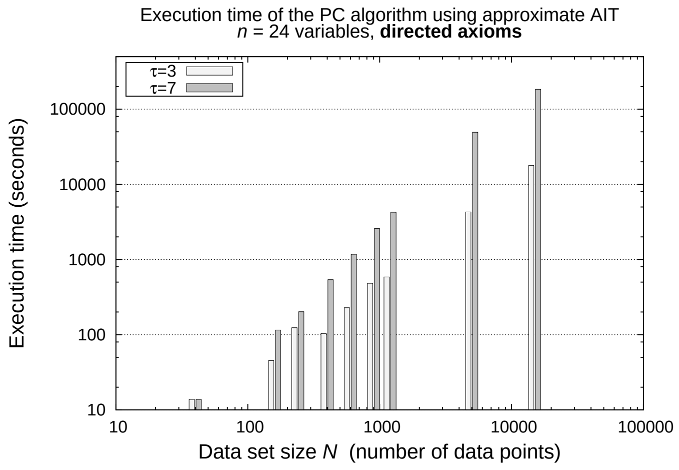

Figure 7: Execution times for the PC algorithm using the approximate argumentative test on the directed axioms ($\widehat{\text{AIT}_t}$-D) on data sets sampled from Bayesian models for domain size $n = 24$ and maximum degrees $\tau = 3, 7$. For the approximate AIT test we limited the depth of the dialog tree to 3 and its the branching factor as described in Section 6.

Figure 6 shows a comparison of the argumentative tests $\widehat{\text{AIT}_t}$-D using the directed axioms with the corresponding SIT. The figure shows two plots for different values of τ of the mean values (over runs for ten different data sets) of $\widehat{acc}^{\text{PC}}_{\text{SIT}}$ and $\widehat{acc}^{\text{PC}}_{\widehat{\text{AIT}_t}\text{-D}}$ (histograms), the difference between these averages (line graph), and the 95% confidence intervals for the difference (error bars), for different data set sizes N. As usual, a positive value of the difference corresponds to an improvement of $\widehat{\mathrm{AIT}}_{\mathrm{t}}$-D over SIT. As in practically all experiments so far, we have statistically significant improvements over the entire range of data set sizes, with maximum improvements of up to 20% for $\tau = 3$, $N = 25000$, and $\tau = 7$, $N = 900$. The corresponding execution times for the entire PC algorithm are shown in Fig. 7. We can make two observations from this graph. One, the cost is significantly lower for sparse domains, which benefits real-world application domains that are sparse. The second observation is that the execution time scales linearly with the number of data points; this exhibits the same behavior as the use of a SIT test in PC, as each test needs to scan the data set once to compute the contingency table and relevant test statistics.

In summary, these results demonstrate that the approximate argumentative test is practical for larger domains and can result in positive, statistically significant accuracy improvements when used for causal discovery. However, the cost of AIT for large data sets, although not prohibitive, can be non-negligible. Therefore the accuracy benefits of AIT vs. a SIT must be carefully weighed off the ability of the user to expend the extra computation. Note that the practicality of the approximate algorithm also depends on the parameters used (the cutoff depth of iterative deepening and the branching factor limit—see Section 6); different parameter values or alternative ways of limiting the size of the dialog tree may be needed for even larger domains.

7.3 Real-world and Benchmark Data Experiments

While the sampled data set studies of the previous section have the advantage of a more controlled and systematic study of the performance of the algorithms, experiments on real-world data are necessary for a more realistic assessment. In this section we present experiments on a number of real-world and benchmark data sets obtained from the UCI machine learning repository (D. J. Newman and Merz, 1998) and the Knowledge Discovery Data repository (Hettich and Bay, 1999). As in the sampled data case of the previous section, for each data set D, we conducted experiments on subsets of D containing an increasing number of data points N to assess the performance of the independence tests on varying conditions of reliability. Again, to reduce variance we repeated each experiment ten times, each time choosing a different randomly selected data subset of equal size.

Because for real-world data sets the underlying model is unknown, we could only be sure the general axioms of Eq. (5) apply. We therefore only used these axioms in this section. Also, as mentioned in the previous section, because some of the data sets have much larger domains (e.g., the alarm data set contains 37 variables), and given the exponential nature of the exact algorithms we could only perform experiments for the approximate version of the argumentative test. For these reasons, in the following experiments we only report the accuracy of $\widehat{\mathrm{AIT}}_{\mathrm{t}}$-G, the approximate argumentative independence test defined over the general axioms. Unfortunately, for real-world data the underlying model is typically unknown and therefore it is impossible to know the true value of any independence. We therefore approximate it by a statistical test on the entire data set, and limit the size of the data set subsets that we use up to a third of the size of the entire data set. This corresponds to the hypothetical scenario that a much smaller data set is available to the researcher, allowing us to evaluate the improvement of argumentation under these more challenging situations. Again, as in the previous two sections, for comparison we sampled 2,000 triplets and calculated the accuracy as a fraction of tests correct, where for the true value of independences and dependences we used the method just described.

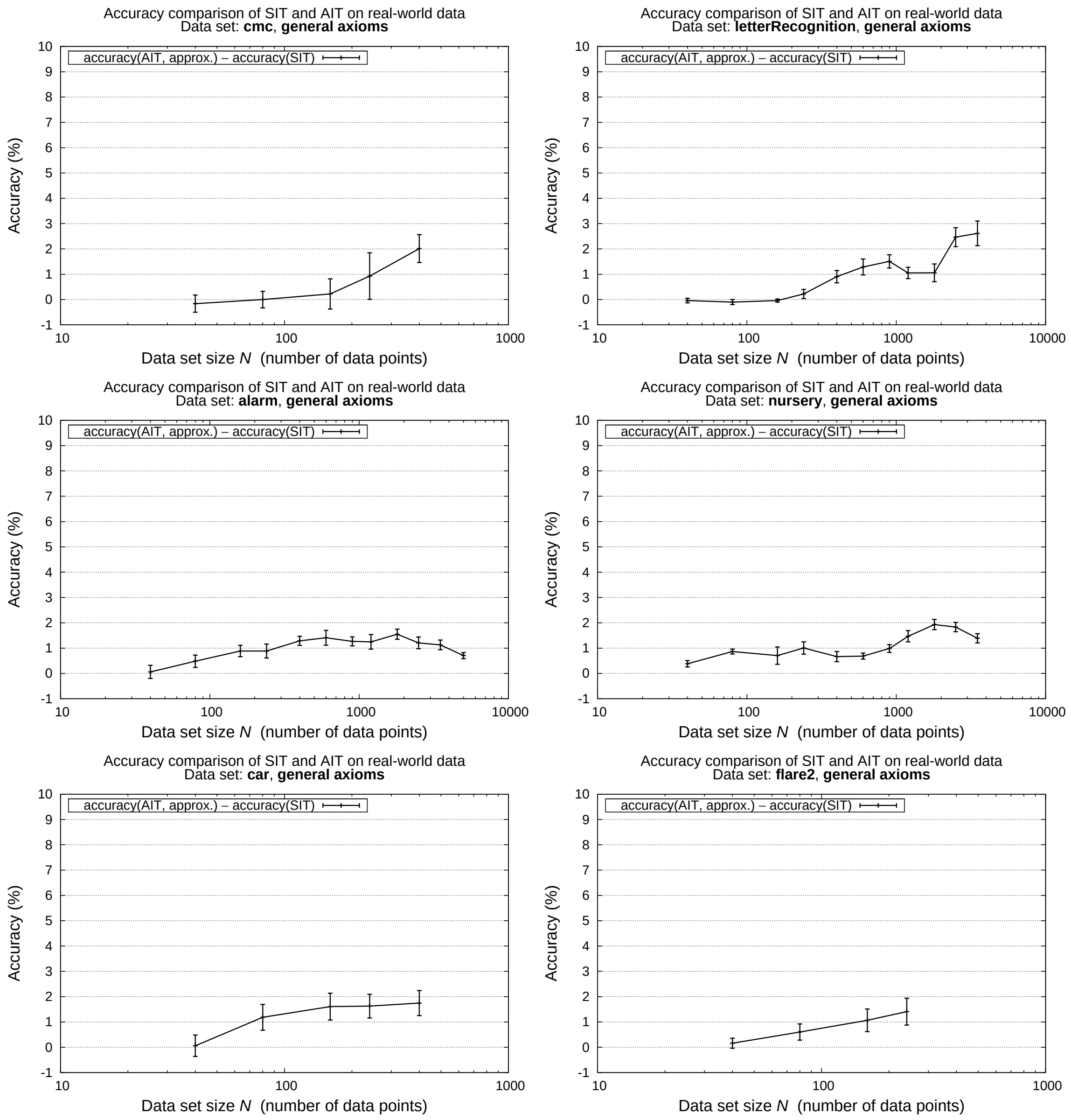

Figure 8: Difference in the mean value of the accuracy $\widehat{\text{AIT}}_t$-G with the mean value of the accuracy of SIT for a number of real-world data sets. The error bars denote the 95% confidence interval of the difference.

Figure 8 and Table 1 show the result of our comparison between the argumentative test $\widehat{\text{AIT}}_t$-G and statistical test SIT for real-world data sets. In the table, the best-performing method is shown in bold. The figure contains 6 plots, one for each data set, depicting the difference between the mean value of the accuracy of $\widehat{\text{AIT}}_t$-G and that of SIT, where as usual a positive value denotes an improvement of $\widehat{\text{AIT}}_t$-G over SIT. While in a few cases the average difference is negative (e.g., data set `cmc`, $N = 40$), in each case the negative value is not statistically significant as the confidence in-

	Data set	`car`	`cmc`	`flare2`	`letterRecognition`	`nursery`	`alarm`
	Domain size	7	10	13	17	9	37
	Data set size	1730	1475	1067	20002	12962	20003
$N = 40$	SIT	80.1	**77.8**	77.0	**47.9**	83.3	76.7
	$\widehat{\text{AIT}}_t$-G	80.1	77.5	**77.1**	47.8	**83.8**	76.7
	$\widehat{\text{AIT}}_t$-G – SIT	0.0 ± 0.7	-0.3 ± 0.6	0.1 ± 0.3	-0.1 ± 0.2	0.4 ± 0.1	0.0 ± 0.4
	Runtime of $\widehat{\text{AIT}}_t$-G (ms)	0.56	1.07	2.61	4.19	0.88	52.06
$N = 240$	SIT	86.7	84.1	85.5	50.7	86.1	84.3
	$\widehat{\text{AIT}}_t$-G	**88.6**	**84.7**	**86.9**	**51.0**	**87.2**	**85.1**
	$\widehat{\text{AIT}}_t$-G – SIT	1.9 ± 0.6	0.5 ± 1.2	1.3 ± 0.8	0.2 ± 0.4	1.1 ± 0.4	0.8 ± 0.3
	Runtime of $\widehat{\text{AIT}}_t$-G (ms)	1.37	5.19	8.73	90.50	1.84	202.05
$N = 600$	SIT				55.8	88.5	88.6
	$\widehat{\text{AIT}}_t$-G				**57.3**	**89.3**	**89.8**
	$\widehat{\text{AIT}}_t$-G – SIT				1.5 ± 0.5	0.8 ± 0.1	1.2 ± 0.4
	Runtime of $\widehat{\text{AIT}}_t$-G (ms)				575.53	4.37	547.77
$N = 1200$	SIT				63.3	89.7	90.8
	$\widehat{\text{AIT}}_t$-G				**64.3**	**91.2**	**92.0**
	$\widehat{\text{AIT}}_t$-G – SIT				1.0 ± 0.3	1.5 ± 0.3	1.2 ± 0.4
	Runtime of $\widehat{\text{AIT}}_t$-G (ms)				2008.76	14.05	1151.05
$N = 3500$	SIT				73.8	94.1	95.2
	$\widehat{\text{AIT}}_t$-G				**76.5**	**95.4**	**96.3**
	$\widehat{\text{AIT}}_t$-G – SIT				2.6 ± 0.7	1.3 ± 0.3	1.1 ± 0.3
	Runtime of $\widehat{\text{AIT}}_t$-G (ms)				24540.51	76.48	3895.2

Table 1: Average accuracies (in percentage) of SIT and $\widehat{\text{AIT}}_t$-G, their differences (denoted $\widehat{\text{AIT}}_t$-G – SIT in the table), the 95% confidence interval for the difference, and the average runtime per test (in ms) for $\widehat{\text{AIT}}_t$-G for several real-world and benchmark data sets. For each data set the table shows these quantities for number of data points $N = 40, 240, 600, 1200, 3500$. The best performing algorithm ($\widehat{\text{AIT}}_t$-G or SIT, with respect to accuracy) is indicated in bold. Empty cells correspond to cases where one third of the data set was smaller than the value of N in that column.

terval contains a portion of the positive half-plane. The figure demonstrates a clear advantage of the argumentative approach, with all data sets reaching statistically significant positive improvements in accuracy of up to 3% and all confidence intervals covering positive values either partially or completely. The table also shows the average execution time (in ms) for the $\widehat{\text{AIT}}_t$-G tests evaluated.

8. Conclusion

We presented a framework for addressing one of the most important problems of independence-based structure discovery algorithms, namely the problem of unreliability of statistical independence tests. Our main idea was to recognize the existence of interdependences among the outcomes of conditional independence tests—in the form of Pearl's axiomatic characterization of the conditional independence relation—that can be seen as integrity constraints and exploited to correct unreliable statistical tests. We modeled this setting as a knowledge base containing conditional independences that are potentially inconsistent, and used the preference-based argumentation framework to reason with and resolve these inconsistencies. We presented in detail how to apply the argumentation framework to independence knowledge bases and how to compute the preference among the independence propositions. We also presented a number of algorithms, both exact and approximate, for implementing statistical testing using this framework. We analyzed the approximate algorithm and proved that is has polynomial worst-case execution time. We also experimentally verified that its accuracy improvement is close to the exact one while providing orders of magnitude faster execution, making possible its use for causal discovery in large domains. Overall, our experimental evaluation demonstrated statistically significant improvements in the accuracy of causal discovery for the overwhelming majority of sampled, benchmark and real-world data sets.

Appendix A. Computability of the Argumentative Independence Test

In this appendix we prove that the argumentative test terminates, a property that we call its *computability*. Some of the theorems and lemmas presented are not original work but adaptations of well known properties of relations. We include them to allow a complete exposition of the proof of computability, given by Theorem 21. We first introduce some notation. We denote independence propositions (e.g., $(X \perp\!\!\!\perp Y \mid \mathbf{Z})$) by σ and their negation (e.g., $(X \not\perp\!\!\!\perp Y \mid \mathbf{Z})$) by $\neg\sigma$. We abbreviate their corresponding propositional arguments $(\{\sigma\}, \sigma)$ and $(\{\neg\sigma\}, \neg\sigma)$ by a_σ and $a_{\neg\sigma}$, respectively, and we will refer to $a_{\neg\sigma}$ as the *negation* of a_σ (and vice versa). Also, we use the predicates $A(a)$, $R(a)$, $Ab(a)$ to denote the fact the argument a is accepted, rejected, or in abeyance, respectively.

For completeness we repeat here the definition of strict and transitive preference relation.

Definition 13. *We say preference relation π over arguments is* strict *if the ordering of arguments induced by it is strict and total, that is, for every pair of arguments a and b,*

$$a \gg_\pi b \iff \neg\big(b \gg_\pi a\big). \tag{27}$$

Definition 14. *We say that preference relation π over arguments is* transitive *if, for every three arguments a, b and c,*

$$\big(a \gg_\pi b\big) \wedge \big(b \gg_\pi c\big) \implies \big(a \gg_\pi c\big).$$

Lemma 24. *A strict preference relation π satisfies the condition that for every pair of arguments such that a defeats b and b defeats a, it is the case that a attacks b or b attacks a, that is, at least one of a and b attacks the other.*

Proof We prove by contradiction: Let us assume that a defeats b and b defeats a but neither a attacks b nor b attacks a. By definition of the attack relation (Definition 15),

$$\neg\big(a \text{ } attacks \text{ } b\big) \implies \neg\big(\neg(b \gg_\pi a)\big) \implies b \gg_\pi a$$

and

$$\neg\big(b \text{ } attacks \text{ } a\big) \implies \neg\big(\neg(a \gg_\pi b)\big) \implies a \gg_\pi b.$$

However, this is a contradiction since, by assumption, the preference ordering is strict, and therefore it cannot be true that both $a \gg_\pi b$ and $b \gg_\pi a$ are true at the same time. ■

Lemma 25. *A strict preference π satisfies the condition that for every pair a and b of arguments, it is not the case that both a attacks b and b attacks a, that is, there can be no mutual attack.*

Proof We prove by contradiction. Let us consider two mutually attacking arguments a and b. By the definition of the attack relation, and because π is a total order, we have that

$$a \text{ } attacks \text{ } b \implies \neg(b \gg_\pi a) \implies \big(a \gg_\pi b \vee a \equiv_\pi b\big)$$

and

$$b \textit{ attacks } b \implies \neg(a \gg_\pi b) \implies \big(b \gg_\pi a \vee b \equiv_\pi a\big)$$

where $a \equiv_\pi b$ means a is equally preferable to b. However, equality of preference is not possible in a strict preference relation. Therefore it must be the case that $a \gg_\pi b$ and $b \gg_\pi a$, which is a contradiction of Eq. (27), again due to strictness. ■

We next prove that no argument is in abeyance if the preference relation over arguments is strict. For that, we first prove that an argument in abeyance is always attacked by at least another argument in abeyance.

Lemma 8. *For every argument a,*

$$Ab(a) \implies \exists b \in attackers(a), Ab(b).$$

Proof By definition, an argument a is in abeyance if it is neither accepted nor rejected. Applying the definitions of acceptance and rejection and manipulating the Boolean formulae we obtain,

$$\begin{aligned}
Ab(a) &\iff \neg A(a) \wedge \neg R(a) \\
&\iff \neg\big(\forall b \in attackers(a), R(b)\big) \wedge \neg\big(\exists b \in attackers(a), A(b))\big) \\
&\iff \big(\exists b \in attackers(a), \neg R(b)\big) \wedge \big(\forall b \in attackers(a), \neg A(b))\big) \\
&\iff \big(\exists b \in attackers(a), (A(b) \vee Ab(b))\big) \wedge \big(\forall b \in attackers(a), \neg A(b))\big) \\
&\iff \big(\exists b \in attackers(a), Ab(b)\big) \wedge \big(\forall b \in attackers(a), \neg A(b))\big) \\
&\implies \exists b \in attackers(a), Ab(b).
\end{aligned}$$

■

Definition 22. *An* attack sequence *is a sequence $\langle a_1, a_2, \ldots, a_n\rangle$ of n arguments such that for every $2 \le i \le n$, a_i attacks a_{i-1}.*

Lemma 26. *Let $\langle \mathcal{A}, \mathcal{R}, \pi\rangle$ be a PAF with a strict and transitive preference relation π. Then, no argument can appear more than once in any attack sequence, that is, for every attack sequence $\langle a_1, a_2, \ldots, a_n\rangle$ and every pair of integers $i, j \in [1,n]$ such that $i \neq j$, $a_i \neq a_j$.*

Proof

We first note that by definition of the attack relation, it must be the case that for any two consecutive arguments a_i, a_{i+1}, it is true that $\neg(a_i \gg_\pi a_{i+1})$. Since π is strict, this is equivalent to $a_{i+1} \gg_\pi a_i$ (c.f. Eq. (27)). That is,

$$a_n \gg_\pi a_{n-1} \gg_\pi \ldots \gg_\pi a_2 \gg_\pi a_1 \tag{28}$$

We now assume, for contradiction, there exists an argument $a^\star$ that appears twice in the attack sequence at indexes $i^\star$ and $j^\star$, that is,

$$\exists\ i^\star, j^\star \in [1,n], i^\star \neq j^\star, \text{ such that } a_{i^\star} = a_{j^\star} = a^\star.$$

Since no argument defeats itself, it cannot attack itself, and thus the smallest possible attack sequence with a repeated argument must have at least length 3. From this fact, Eq. (28), and transitivity, there must exist an argument $b \neq a^\star$ such that $a^\star \gg_\pi b \gg_\pi a^\star$. This last fact implies that $a^\star \gg_\pi b$ and $b \gg_\pi a^\star$ must hold, which contradicts strictness (Eq. (27)).

■

A corollary of this lemma is the following theorem.

Theorem 23. *Every attack sequence $\langle a_1, a_2, \ldots, a_n \rangle$ in a PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ with strict and transitive π, and finite $\mathcal{A}$ is finite.*

Proof Follows directly from Lemma 26 and the fact that $\mathcal{A}$ is finite. ■

We can now prove the main result of this section in the following theorem.

Theorem 21. *Given an arbitrary triplet $t = (X, Y \mid \mathbf{Z})$, and a PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ with a strict and transitive preference relation π, and finite arguments set $\mathcal{A}$, the top-down algorithm of Algorithm 3 run for input t over $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ terminates.*

Proof In the tree traversed by the top-down algorithm, any path from the root to a leaf is an attack sequence. Since for strict and transitive π, and finite $\mathcal{A}$ each such sequence is finite, the algorithm always terminates. ■

Appendix B. Validity of the Argumentative Independence Test

In this section we prove the property of the argumentative independence test of deciding that an input triplet $(X, Y \mid \mathbf{Z})$ evaluates to either independence or dependence, but not both or neither. We call this property the *validity* of the test.

We start we proving that under the assumption of a strict and transitive preference relation, no argument is in abeyance.

Theorem 20. *Let $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ be a PAF with a strict and transitive preference relation π. Then no argument $a \in \mathcal{A}$ is in abeyance.*

Proof Let us assume, for contradiction, that there is an argument a in abeyance. From Lemma 8, not only a has an attacker in abeyance, say argument b, but b also has an attacker in abeyance, and so on. That is, we can construct an attack sequence starting at a that contains only arguments in abeyance. Moreover, this sequence must be infinite, since the lemma assures as we always have at least one attacker in abeyance. This is in direct contradiction with Theorem 23. ■

Corollary 27. *For every argument a in a PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ with strict and transitive π,*

$$A(a) \iff \neg R(a).$$

We now prove a number of lemmas that hold only for the sub-class of propositional arguments (arguments whose support contains only one proposition, equal to the head of that argument). We start with a lemma that demonstrates that it cannot be the case that an attacker of a propositional argument a_σ and an attacker of its negation $a_{\neg\sigma}$ do not attack each other. The former must attack the latter or vice versa.

Lemma 28. *Let $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ be a PAF with a strict preference relation π, $a_\sigma \in \mathcal{A}$ be a propositional argument, and $a_{\neg\sigma}$ its negation. For every pair of arguments b and c that attacks a_σ and $a_{\neg\sigma}$ respectively,*

$$(b \textit{ attacks } c) \vee (c \textit{ attacks } b).$$

Proof Since a_σ and $a_{\neg\sigma}$ are propositional arguments, their support contains the head and only the head, and thus any defeater (i.e., rebutter or undercutter) must have as head $\neg\sigma$ and σ, respectively, that is, the head of b must be $\neg\sigma$ and the head of c must be σ. Thus, b rebuts (and thus defeats) c and vice versa. The lemma then follows directly from Lemma 24. ■

Lemma 29. *Let $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ be a PAF with a strict preference relation π, and a_σ and $a_{\neg\sigma}$ be a propositional argument and its negation. Then,*

$$R(a_\sigma) \implies \neg R(a_{\neg\sigma}).$$

Proof By assumption, $R(a_\sigma)$. We assume, for contradiction, that $R(a_{\neg\sigma})$. Therefore, by the definition of rejection, $\exists b \in attackers(a_\sigma)$ such that $A(b)$, and $\exists c \in attackers(a_{\neg\sigma})$ such that $A(c)$. By Lemma 28 b *attacks* c or c *attacks* b. In either case, an accepted argument is attacking an accepted argument, which contradicts the definition of acceptance. ■

Lemma 30. *Given a PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ with a strict preference relation π, every propositional argument $a_\sigma \in \mathcal{A}$ satisfies*

$$A(a_\sigma) \implies \neg A(a_{\neg\sigma})$$

Proof We prove by contradiction. Let us assume that both a_σ and $a_{\neg\sigma}$ are accepted. Since a_σ and $a_{\neg\sigma}$ are propositional arguments, they defeat each other. Then, by Lemma 24 a_σ attacks $a_{\neg\sigma}$ or vice versa. In either case an accepted argument has an accepted attacker, which is a contradiction. ■

We now prove Theorem 19 that was introduced in Section 4, reproduced here for convenience.

Theorem 19. *Given a PAF $\langle \mathcal{A}, \mathcal{R}, \pi \rangle$ with a strict and transitive preference relation π, every propositional argument $a_\sigma \in \mathcal{A}$ and its negation $a_{\neg\sigma}$ satisfy*

$$A(a_\sigma) \iff R(a_{\neg\sigma}).$$

Proof The ($\implies$) direction follows from Lemma 30 and Theorem 20. The ($\Longleftarrow$) direction follows from Lemma 29 and Theorem 20. ■

In Section 4 we defined the following semantics for deciding on the dependence or independence of an input triplet $(X, Y \mid \mathbf{Z})$:

$$\begin{aligned} (\{(X \not\perp\!\!\!\perp Y \mid \mathbf{Z})\}, (X \not\perp\!\!\!\perp Y \mid \mathbf{Z})) \text{ is accepted} &\iff (X \not\perp\!\!\!\perp Y \mid \mathbf{Z}) \text{ is accepted} \implies (X \not\perp\!\!\!\perp Y \mid \mathbf{Z}) \\ (\{(X \perp\!\!\!\perp Y \mid \mathbf{Z})\}, (X \perp\!\!\!\perp Y \mid \mathbf{Z})) \text{ is accepted} &\iff (X \perp\!\!\!\perp Y \mid \mathbf{Z}) \text{ is accepted} \implies (X \perp\!\!\!\perp Y \mid \mathbf{Z}) \qquad (29) \end{aligned}$$

where acceptance is defined over an independence-based PAF as defined in Section 3.3. For this argumentative test of independence to be valid, its decision must be non-ambiguous, that is, it must decide either independence or dependence, but not both or neither. For that, exactly one of the antecedents of the above implications must be true. Formally:

Theorem 18. *For any input triplet $\sigma = (X, Y \mid \mathbf{Z})$, the argumentative independence test defined by Eq. (29) produces a non-ambiguous decision, that is, it decides that σ evaluates to either independence or dependence, but not both or neither.*

Proof Let us denote $(X \perp\!\!\!\perp Y \mid \mathbf{Z})$ by σ_{t} and $(X \not\perp\!\!\!\perp Y \mid \mathbf{Z})$ by σ_{f}. Since strictness and transitivity of the independence preference relation hold (proved in Section 3.3, lemmas 16 and 17 respectively), Theorems 19 and 20 hold as well. From Theorem 20 we know that neither of the propositional arguments is in abeyance. Thus, since $a_{\sigma_{\mathsf{t}}}$ corresponds to the negation of $a_{\sigma_{\mathsf{f}}}$ it follows from Theorem 19 that exactly one of them is accepted. ■

References

H. Abdi. The Bonferonni and Šidák corrections for multiple comparisons. In Neil Salkind, editor, *Encyclopedia of Measurement and Statistics*. Thousand Oaks (CA): Sage, 2007.

A. Agresti. *Categorical Data Analysis*. Wiley, 2nd edition, 2002.

L. Amgoud and C. Cayrol. A reasoning model based on the production of acceptable arguments. *Annals of Mathematics and Artificial Intelligence*, 34:197–215, 2002.

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B (Methodological)*, 57 (1):289–300, 1995.

Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, 29(4):1165–1188, 2001.

W. G. Cochran. Some methods of strengthening the common χ^2 tests. *Biometrics*, 10:417–451, 1954.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

C. L. Blake D. J. Newman, S. Hettich and C. J. Merz. UCI repository of machine learning databases. *Irvine, CA: University of California, Department of Information and Computer Science*, 1998.

A. P. Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society*, 41:1–31, 1979.

P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and *n*-person games. *Artificial Intelligence*, 77:321–357, 1995.

P. Gärdenfors. *Belief Revision*. Cambridge Computer Tracts. Cambridge University Press, Cambridge, 1992.

P. Gärdenfors and H. Rott. Belief revision. In Gabbay, D. M., Hogger, C. J. and Robinson, J. A., editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4. Clarendon Press, Oxford, 1995.

S. Hettich and S. D. Bay. The UCI KDD archive. *Irvine, CA: University of California, Department of Information and Computer Science*, 1999.

Y. Hochberg. A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*, 75(4): 800–802, December 1988.

J. S. Ide, F. G. Cozman, and F. T. Ramos. Generating random Bayesian networks with constraints on induced width. *Brazilian Symposium on Artificial Intelligence, Recife, Pernambuco, Brazil*, 2002.

A. C. Kakas and F. Toni. Computing argumentation in logic programming. *Journal of Logic and Computation*, 9(4):515–562, 1999.

M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, 1994.

R. P. Loui. Defeat among arguments: a system of defeasible inference. *Computational Intelligence*, 2:100–106, 1987.

J. P. Martins. Belief revision. In Shapiro, S. C., editor, *Encyclopedia of Artificial Intelligence*, pages 110–116. John Wiley & Sons, New York, second edition, 1992.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, 2nd edition, 1988.

J. Pearl and A. Paz. GRAPHOIDS: A graph-based logic for reasoning about relevance relations. Technical Report 850038 (R-53-L), Cognitive Systems Laboratory, University of California, 1985.

J. L. Pollock. How to reason defeasibly. *Artificial Intelligence*, 57:1–42, 1992.

H. Prakken. *Logical Tools for Modelling Legal Argument. A Study of Defeasible Reasoning in Law.* Kluwer Law and Philosophy Library, Dordrecht, 1997.

H. Prakken and G. Vreeswijk. *Logics for Defeasible Argumentation*, volume 4 of *Handbook of Philosophical Logic*. Kluwer Academic Publishers, Dordrecht, 2 edition, 2002.

S. C. Shapiro. Belief revision and truth maintenance systems: An overview and a proposal. Technical Report CSE 98-10, Dept of Computer Science and Engineering, State University of New York at Buffalo, 1998.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. Adaptive Computation and Machine Learning Series. MIT Press, 2nd edition, January 2000.

J. D. Storey. A direct approach to false discovery rates. *Journal of the Royal Statistical Society, Series B (Methodological)*, 64(3):479–498, 2002.

M. Studený. Conditional independence relations have no finite complete characterization. In *Transactions of the 11th Prague Conference on Information Theory, Statistical Decision Functions and Random Processes*, volume B, pages 377–396, 1991.

F. Toni and A. C. Kakas. In A. Nerode, editor, *3rd International Conference on Logic Programming and Non-monotonic Reasoning*, volume 928 of *Lecture Notes in Artificial Intelligence*, pages 401–415. Springer Verlag, 1995.